

AD-A282 863



ADST/WDL/TR--93-003036

**ADST**  
**Software Design Document, Volume III**  
**AIRNET**  
**Digital Message Communications Console**  
**(CSCI 11)**

Loral Western Development Labs  
Electronic Defense Systems Software Department  
Software Engineering Laboratory  
3200 Zanker Road  
San Jose, California 95161-9041

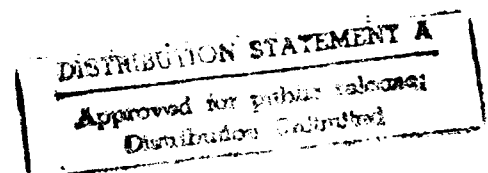
March 31, 1993

Contract No. N61339-91-D-0001  
CDRL A009

Prepared for

Simulation Training and Instrumentation Command  
Naval Training Systems Center  
12350 Research Parkway  
Orlando, FL 32826-3275

**LORAL**  
Western Development Labs



94-24940



DTIC QUALITY INSPECTED 1

94 8 08 018

March 31, 1993  
February 19, 1993

<b>REPORT DOCUMENTATION PAGE</b>			Form approved OMB No.	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing the burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 31 MARCH 1993		3. REPORT TYPE AND DATES COVERED Version 1.0
4. TITLE AND SUBTITLE ADST Software Design Document: Volume III: AIRNET Digital Message Communications Console			5. FUNDING NUMBERS Contract No. N61339-91-D-0001	
6. AUTHOR(S) Aiken, John; Desmeules, Peter; Au-Yeung, Anna; McNerney, Tim; Stephan, Jeff				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Loral Systems Company ADST Program Office 12443 Research Parkway, Suite 303 Orlando, FL 32826			8. PERFORMING ORGANIZATION REPORT NUMBER ADST/WDL/TR-93-003036	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Simulation, Training and Instrumentation Command STRICOM Naval Training Systems Center 12350 Research Parkway Orlando, FL 32826-3275			10. SPONSORING ORGANIZATION REPORT	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Final version for public release.			12b. DISTRIBUTION CODE  A	
13. ABSTRACT (Maximum 200 words)  The Software Design Document describes the design of the AIRNET Digital Message Communications software.				
14. SUBJECT TERMS			15. NUMBER OF PAGES 629	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	17. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	17. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	18. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std Z39-18  
298-102

1.	Scope.....	1
1.1	Identification.....	1
1.2	System overview.....	1
1.3	Document overview.....	4
2.	Referenced and Relevant documents.....	6
2.1	Government documents.....	6
2.2	Non-Government documents.....	6
3.	Preliminary design.....	8
3.1	CSCI overview.....	8
3.1.1	CSCI architecture.....	9
3.1.2	System states and modes.....	18
3.1.3	Memory and processing time allocation.....	19
3.2	CSCI design description.....	20
3.2.1	CSC dms.....	20
3.2.2	CSC build pdu.....	23
3.2.3	CSC ipc.....	23
3.2.4	CSC ethernet.....	25
3.2.4.1	Sub-level CSC dmcc_sim_rx process.....	26
3.2.4.2	Sub-level CSC dmcc_sim_tx.....	26
3.2.5	CSC client.....	27
3.2.6	CSC pdu_decode.....	28
4.	Detailed design.....	30
4.1	CSC dms.....	31
4.1.1	CSU main.....	32
4.1.1.1	CSU main design specification/constraints.....	32
4.1.1.2	CSU main design.....	32
4.1.2	CSU init_queue.....	33
4.1.2.1	CSU init_queue requirements.....	33
4.1.2.2	CSU init_queue design.....	33
4.1.3.	CSU init_shared_mem.....	34
4.1.3.1	CSU init_shared_mem requirements.....	34
4.1.3.2	CSU init_shared_mem design.....	34
4.1.4.	CSU wait_for_msg.....	35
4.1.4.1	CSU wait_for_msg requirements.....	35
4.1.4.2	CSU wait_for_msg design.....	35
4.1.5	CSU cmpfixedstr.....	36
4.1.5.1	CSU cmpfixedstr requirements.....	36
4.1.5.2	CSU cmpfixedstr design.....	37
4.1.6	CSU trimup.....	38
4.1.6.1	CSU trimup requirements.....	38
4.1.6.2	CSU trimup design.....	38
4.1.7	CSU cleanup_xchild.....	39
4.1.7.1	CSU cleanup_xchild requirements.....	39
4.1.7.2	CSU cleanup_xchild design.....	39
4.2.	CSC build pdu.....	40
4.2.1	CSU setdtg.....	42

Dist	Approved
A-1	

4.2.1.1	CSU setdtg requirements.....	42
4.2.1.2	CSU setdtg design.....	42
4.2.2	CSU bldAck.....	43
4.2.2.1	CSU bldAck requirements.....	43
4.2.2.2	CSU bldAck design.....	43
4.2.3	CSU bldFreeText.....	44
4.2.3.1	CSU bldFreeText requirements.....	44
4.2.3.2	CSU bldFreeText design.....	44
4.2.4	CSU bldSpot.....	45
4.2.4.1	CSU bldSpot requirements.....	45
4.2.4.2	CSU bldSpot design.....	45
4.2.5	CSU bldRequest.....	46
4.2.5.1	CSU bldRequest requirements.....	46
4.2.5.2	CSU bldRequest design.....	47
4.2.6	CSU bldMove.....	47
4.2.6.1	CSU bldMove requirements.....	48
4.2.6.2	CSU bldMove design.....	48
4.2.7	CSU bldMTO.....	49
4.2.7.1	CSU bldMTO requirements.....	49
4.2.7.2	CSU bldMTO design.....	49
4.2.8	CSU bldShot.....	50
4.2.8.1	CSU bldShot requirements.....	50
4.2.8.2	CSU bldShot design.....	50
4.2.9	CSU bldSplash.....	51
4.2.9.1	CSU bldSplash requirements.....	51
4.2.9.2	CSU bldSplash design.....	51
4.2.10	CSU fwdpdu.....	52
4.2.10.1	CSU fwdpdu requirements.....	53
4.2.10.2	CSU fwdpdu design.....	53
4.3	CSC ipc.....	54
4.3.1	CSU dmsinit.....	55
4.3.1.1	CSU dmsinit requirements.....	55
4.3.1.2	CSU dmsinit design.....	55
4.3.2	CSU dms_senddm.....	56
4.3.2.1	CSU dms_senddm requirements.....	56
4.3.2.2	CSU dms_senddm design.....	56
4.3.3	CSU dms_recvdm.....	57
4.3.3.1	CSU dms_recvdm requirements.....	57
4.3.3.2	CSU dms_recvdm design.....	57
4.3.4	CSU dms_getfreerecord.....	58
4.3.4.1	CSU dms_getfreerecord requirements.....	58
4.3.4.2	CSU dms_getfreerecord design.....	58
4.3.5	CSU dms_seekrecord.....	59
4.3.5.1	CSU dms_seekrecord requirements.....	59
4.3.5.2	CSU dms_seekrecord design.....	60
4.3.6	CSU dms_notify.....	60



4.3.6.1	CSU dms_notify requirements.....	60
4.3.6.2	CSU dms_notify design.....	61
4.3.7	CSU dms_bcopy.....	62
4.3.7.1	CSU dms_bcopy requirements.....	62
4.3.7.2	CSU dms_bcopy design.....	62
4.3.8	CSU dmslogin.....	63
4.3.8.1	CSU dmslogin requirements.....	63
4.3.8.2	CSU dmslogin design.....	63
4.3.9	CSU dmsnewdests.....	64
4.3.9.1	CSU dmsnewdests requirements.....	64
4.3.9.2	CSU dmsnewdests design.....	64
4.3.10.	CSU dmslogout.....	65
4.3.10.1	CSU dmslogout requirements.....	65
4.3.10.2	CSU dmslogout design.....	65
4.3.11	CSU dms_getdm.....	66
4.3.11.1.	CSU dms_getdm requirements.....	66
4.3.11.2	CSU dms_getdm design.....	67
4.3.12	CSU dms_getexid.....	67
4.3.12.1	CSU dms_getexid requirements.....	68
4.3.12.2	CSU dms_getexid design.....	68
4.3.13	CSU dms_getopername.....	69
4.3.13.1	CSU dms_getopername requirements.....	69
4.3.13.2	CSU dms_getopername design.....	69
4.4.	CSC Ethernet.....	70
4.4.1.	Sub-Level CSC dmcc_sim_rx.c.....	70
4.4.1.1	CSU main.....	72
4.4.1.1.1	CSU main requirements.....	72
4.4.1.1.2.	CSU main design.....	72
4.4.1.2	CSU SIM_rx_netif.....	73
4.4.1.2.1	CSU SIM_rx_netif requirements.....	73
4.4.1.2.2.	CSU SIM_rx_netif design.....	74
4.4.1.3	CSU ALPDU_SIM_decode.....	77
4.4.1.3.1	CSU ALPDU_SIM_decode requirements.....	77
4.4.1.3.2.	CSU ALPDU_SIM_decode design.....	77
4.4.1.4	CSU check_exercise_ID_SIM.....	79
4.4.1.4.1	CSU check_exercise_ID_SIM requirements.....	80
4.4.1.4.2.	CSU check_exercise_ID_SIM design.....	80
4.4.1.5	CSU check_SIM_ALPDU.....	81
4.4.1.5.1	CSU check_SIM_ALPDU design requirements.....	81
4.4.1.5.2.	CSU check_SIM_ALPDU design.....	81
4.4.1.6	CSU Rcv_SIM_ALPDU.....	83
4.4.1.6.1	CSU Rcv_SIM_ALPDU design requirements.....	83
4.4.1.6.2	CSU Rcv_SIM_ALPDU design.....	83
4.4.1.7	CSU dms_decode_SIM_pkt.....	85
4.4.1.7.1	CSU dms_decode_SIM_pkt design requirements.....	86
4.4.1.7.2.	CSU dms_decode_SIM_pkt design.....	86

4.4.1.8	CSU logger.....	87
4.4.1.8.1	CSU logger design requirements.....	88
4.4.1.8.2	CSU logger design.....	88
4.4.1.9	CSU ether_open.....	90
4.4.1.9.1	CSU ether_open design requirements.....	90
4.4.1.9.2	CSU ether_open design.....	90
4.4.1.10	CSU niocctl.....	94
4.4.1.10.1	niocctl CSU design requirements.....	94
4.4.1.10.2	niocctl CSU design.....	94
4.4.1.11	CSU ether_interfaces.....	95
4.4.1.11.1	CSU ether_interfaces design requirements.....	96
4.4.1.11.2	CSU ether_interfaces design.....	96
4.4.1.12	CSU shutdown_SIM_rx.....	98
4.4.1.12.1	CSU shutdown_SIM_rx design requirements.....	98
4.4.1.12.2	CSU shutdown_SIM_rx design.....	98
4.4.1.13	CSU dmsinit.....	100
4.4.1.14	CSU dms_bcopy.....	100
4.4.2	Sub-Level CSC dmcc_sim_tx.c.....	100
4.4.2.1	CSU main.....	102
4.4.2.1.1	CSU main design requirements.....	102
4.4.2.1.2	CSU main design.....	102
4.4.2.2	CSU ALPDU_SIM_encode.....	104
4.4.2.2.1	CSU ALPDU_SIM_encode requirements.....	104
4.4.2.2.2	CSU ALPDU_SIM_encode design.....	104
4.4.2.3	CSU SIM_tx_netif.....	106
4.4.2.3.1	CSU SIM_tx_netif requirements.....	106
4.4.2.3.2	CSU SIM_tx_netif design.....	106
4.4.2.4	CSU dmsinit.....	109
4.4.2.5	CSU dms_recvdma.....	109
4.4.2.6	CSU alt_pause.....	109
4.4.2.6.1	CSU pause requirements.....	110
4.4.2.6.2	CSU pause design.....	110
4.4.2.7	CSU alt_flush.....	111
4.4.2.7.1	CSU alt_flush requirements.....	111
4.4.2.7.2	CSU alt_flush design.....	111
4.4.2.8	CSU logger.....	112
4.4.2.9	CSU ether_open.....	113
4.4.2.10	CSU niocctl.....	113
4.4.2.11	CSU ether_interfaces.....	113
4.5	CSC client.....	113
4.5.1	Sub-Level CSC addrListCB.....	114
4.5.1.1	CSU AddrListAddCB.....	114
4.5.1.1.1	CSU AddrListAddCB requirements.....	115
4.5.1.1.2	CSU AddrListAddCB design.....	115
4.5.1.2	CSU AddrListDeleteCB.....	116
4.5.1.2.1	CSU AddrListDeleteCB requirements.....	116

4.5.1.2.2	CSU AddrListDeleteCB design.....	116
4.5.1.3	CSU AddrListSaveExitCB.....	117
4.5.1.3.1	CSU AddrListSaveExitCB requirements.....	117
4.5.1.3.2	CSU AddrListSaveExitCB design.....	117
4.5.1.4	CSU AddrListClearExitCB.....	118
4.5.1.4.1	CSU AddrListClearExitCB requirements.....	118
4.5.1.4.2	CSU AddrListClearExitCB design.....	118
4.5.1.5	CSU PutupAddrList.....	119
4.5.1.5.1	CSU PutupAddrList requirements.....	119
4.5.1.5.2	CSU PutupAddrList design.....	119
4.5.2	Sub-Level CSC consoleCB.....	120
4.5.2.1	CSU ConsoleSysMainBtnCB.....	120
4.5.2.1.1	CSU ConsoleSysMainBtnCB requirements.....	120
4.5.2.1.2	CSU ConsoleSysMainBtnCB design.....	121
4.5.3	Sub-Level CSC freeTxtCB.....	122
4.5.3.1	CSU FreeTxtSendCB.....	122
4.5.3.1.1	CSU FreeTxtSendCB requirements.....	122
4.5.3.1.2	CSU FreeTxtSendCB design.....	122
4.5.3.2	CSU FreeTxtSaveReturn.....	123
4.5.3.2.1	CSU FreeTxtSaveReturn requirements.....	123
4.5.3.2.2	CSU FreeTxtSaveReturn design.....	123
4.5.3.3	CSU FreeTxtClearReturn.....	124
4.5.3.3.1	CSU FreeTxtClearReturn requirements.....	124
4.5.3.3.2	CSU FreeTxtClearReturn design.....	124
4.5.3.4	CSU freetxtad_moreCB.....	125
4.5.3.4.1	CSU freetxtad_moreCB requirements.....	125
4.5.3.4.2	CSU freetxtad_moreCB design.....	126
4.5.3.5	CSU freetxtad_rotateCB.....	126
4.5.3.5.1	CSU freetxtad_rotateCB requirements.....	126
4.5.3.5.2	CSU freetxtad_rotateCB design.....	127
4.5.3.6	CSU PutupFreeTxt.....	128
4.5.3.6.1	CSU PutupFreeTxt requirements.....	128
4.5.3.6.2	CSU PutupFreeTxt design.....	128
4.5.3.7	CSU ClearFreeTxt.....	129
4.5.3.7.1	CSU ClearFreeTxt requirements.....	129
4.5.3.7.2	CSU ClearFreeTxt design.....	129
4.5.4	Sub-Level CSC getMessageCB.....	130
4.5.4.1	CSU getIncomingMsg.....	130
4.5.4.1.1	CSU getIncomingMsg requirements.....	130
4.5.4.1.2	CSU getIncomingMsg design.....	130
4.5.5	Sub-Level CSC groupListCB.....	131
4.5.5.1	CSU GrpListAddCB requirements.....	132
4.5.5.1.1	CSU GrpListAddCB design specification/constraints.....	132
4.5.5.1.2	CSU GrpListAddCB design.....	132
4.5.5.2	CSU GrpListDeleteCB.....	133
4.5.5.2.1	CSU GrpListDeleteCB requirements.....	133

4.5.5.2.2	CSU GrpListDeleteCB design.....	133
4.5.5.3	CSU GrpListSaveExitCB .....	134
4.5.5.3.1	CSU GrpListSaveExitCB requirements. ....	134
4.5.5.3.2	CSU GrpListSaveExitCB design .....	134
4.5.5.4	CSU GrpListClearExitCB.....	135
4.5.5.4.1	CSU GrpListClearExitCB requirements.....	135
4.5.5.4.2	CSU GrpListClearExitCB design.....	135
4.5.5.5	CSU PutupGroupList .....	136
4.5.5.5.1	CSU PutupGroupList design requirements. ....	136
4.5.5.5.2	CSU PutupGroupList design .....	136
4.5.6	Sub-Level CSC listUtilities.....	137
4.5.6.1	CSU ListUpCB.....	137
4.5.6.1.1	CSU ListUpCB requirements.....	138
4.5.6.1.2	CSU ListUpCB design.....	138
4.5.6.2	CSU ListDownCB.....	139
4.5.6.2.1	CSU ListDownCB requirements.....	139
4.5.6.2.2	CSU ListDownCB design .....	139
4.5.6.3	CSU ListAddItem.....	140
4.5.6.3.1	CSU ListAddItem requirements.....	140
4.5.6.3.2	CSU ListAddItem design .....	140
4.5.6.4	CSU ListDeleteItem .....	141
4.5.6.4.1	CSU ListDeleteItem requirements. ....	141
4.5.6.4.2	CSU ListDeleteItem design .....	141
4.5.7	Sub-Level CSC locListCB.....	142
4.5.7.1	CSU LocListAddCB.....	142
4.5.7.1.1	CSU LocListAddCB requirements.....	142
4.5.7.1.2	CSU LocListAddCB design .....	143
4.5.7.2	CSU LocListDeleteCB .....	144
4.5.7.2.1	CSU LocListDeleteCB requirements. ....	144
4.5.7.2.2	CSU LocListDeleteCB design .....	144
4.5.7.3	CSU LocListSaveExitCB.....	145
4.5.7.3.1	CSU LocListSaveExitCB requirements.....	145
4.5.7.3.2	CSU LocListSaveExitCB design .....	145
4.5.7.4	CSU LocListClearExitCB .....	146
4.5.7.4.1	CSU LocListClearExitCB requirements. ....	146
4.5.7.4.2	CSU LocListClearExitCB design .....	146
4.5.7.5	CSU PutupLocationList .....	147
4.5.7.5.1	CSU PutupLocationList requirements. ....	147
4.5.7.5.2	CSU PutupLocationList design .....	147
4.5.8	Sub-Level CSC logonCB. ....	148
4.5.8.1	CSU LogonNetLogonCB.....	148
4.5.8.1.1	CSU LogonNetLogonCB requirements.....	148
4.5.8.1.2	CSU LogonNetLogonCB design.....	148
4.5.8.2	CSU PutupLogon .....	150
4.5.8.2.1	CSU PutupLogon requirements. ....	150
4.5.8.2.2	CSU PutupLogon design .....	150

4.5.9	Sub-Level CSC main.....	151
4.5.9.1	CSU main.....	151
4.5.9.1.1	CSU main requirements.....	151
4.5.9.1.2	CSU main design.....	151
4.5.10	Sub-Level CSC movcmdTmi_call.....	152
4.5.10.1	CSU MovcmdClearNRetCB.....	152
4.5.10.1.1	CSU MovcmdClearNRetCB requirements.....	152
4.5.10.1.2	CSU MovcmdClearNRetCB design.....	152
4.5.10.2	CSU MovcmdSaveNRetCB.....	153
4.5.10.2.1	CSU MovcmdSaveNRetCB requirements.....	154
4.5.10.2.2	CSU MovcmdSaveNRetCB design.....	154
4.5.10.3	CSU MovcmdSndRoutCB.....	155
4.5.10.3.1	CSU MovcmdSndRoutCB requirements.....	155
4.5.10.3.2	CSU MovcmdSndRoutCB design.....	155
4.5.10.4	CSU MovcmdSend.....	156
4.5.10.4.1	CSU MovcmdSend requirements.....	156
4.5.10.4.2	CSU MovcmdSend design.....	156
4.5.10.5	CSU MovcmdSndUrgCB design specification/constraints.....	157
4.5.10.5.1	CSU MovcmdSndUrgCB requirements.....	157
4.5.10.5.2	CSU MovcmdSndUrgCB design.....	157
4.5.10.6	CSU ClearMovcmd.....	158
4.5.10.6.1	CSU ClearMovcmd requirements.....	158
4.5.10.6.2	CSU ClearMovcmd design.....	159
4.5.11	Sub-Level CSC movcmd_call.....	160
4.5.11.1	CSU lctn_moreCB.....	160
4.5.11.1.1	CSU lctn_moreCB requirements.....	160
4.5.11.1.2	CSU lctn_moreCB design.....	160
4.5.11.2	CSU lctn_rotateCB.....	161
4.5.11.2.1	CSU lctn_rotateCB requirements.....	161
4.5.11.2.2	CSU lctn_rotateCB design.....	161
4.5.11.3	CSU task_moreCB.....	162
4.5.11.3.1	CSU task_moreCB requirements.....	162
4.5.11.3.2	CSU task_moreCB design.....	162
4.5.11.4	CSU task_rotateCB.....	163
4.5.11.4.1	CSU task_rotateCB requirements.....	163
4.5.11.4.2	CSU task_rotateCB design.....	164
4.5.11.5	CSU when_rotateCB.....	164
4.5.11.5.1	CSU when_rotateCB requirements.....	165
4.5.11.5.2	CSU when_rotateCB design.....	165
4.5.11.6	CSU movcmdad_moreCB.....	166
4.5.11.6.1	CSU movcmdad_moreCB requirements.....	166
4.5.11.6.2	CSU movcmdad_moreCB design.....	166
4.5.11.7	CSU movcmdad_rotateCB.....	167
4.5.11.7.1	CSU movcmdad_rotateCB requirements.....	167
4.5.11.7.2	CSU movcmdad_rotateCB design.....	167
4.5.11.8	CSU PutupMovcmd.....	168

4.5.11.8.1	CSU PutupMovcmd requirements.....	168
4.5.11.8.2	CSU PutupMovcmd design.....	168
4.5.12	Sub-Level CSC msg1CB.....	169
4.5.12.1	CSU MsgsPrevCB.....	169
4.5.12.1.1	CSU MsgsPrevCB requirements.....	169
4.5.12.1.2	CSU MsgsPrevCB design.....	169
4.5.12.2	CSU MsgsNextCB.....	170
4.5.12.2.1	CSU MsgsNextCB requirements.....	170
4.5.12.2.2	CSU MsgsNextCB design.....	170
4.5.12.3	CSU MsgsReadCB.....	171
4.5.12.3.1	CSU MsgsReadCB requirements.....	171
4.5.12.3.2	CSU MsgsReadCB design.....	171
4.5.12.4	CSU MsgsDeleteCB.....	172
4.5.12.4.1	CSU MsgsDeleteCB requirements.....	173
4.5.12.4.2	CSU MsgsDeleteCB design.....	173
4.5.12.5	CSU FormatMessageInfo.....	174
4.5.12.5.1	CSU FormatMessageInfo requirements.....	174
4.5.12.5.2	CSU FormatMessageInfo design.....	174
4.5.12.6	CSU PutupMsg1.....	175
4.5.12.6.1	CSU MsgsPrevCB requirements.....	175
4.5.12.6.2	CSU MsgsPrevCB design.....	175
4.5.12.7	CSU TestPriority.....	176
4.5.12.7.1	CSU TestPriority requirements.....	176
4.5.12.7.2	CSU TestPriority design.....	176
4.5.12.8	CSU UpdateList.....	177
4.5.12.8.1	CSU UpdateList requirements.....	177
4.5.12.8.2	CSU UpdateList design.....	177
4.5.12.9	CSU MsgsReuseCB.....	178
4.5.12.9.1	CSU MsgsReuseCB requirements.....	179
4.5.12.9.2	CSU MsgsReuseCB design.....	179
4.5.12.10	CSU MsgsReuseNIncludeCB.....	180
4.5.12.10.1	CSU MsgsReuseNIncludeCB requirements.....	180
4.5.12.10.2	CSU MsgsReuseNIncludeCB design.....	180
4.5.12.11	CSU MsgsSaECB.....	181
4.5.12.11.1	CSU MsgsSaECB requirements.....	181
4.5.12.11.2	CSU MsgsSaECB design.....	181
4.5.12.12	CSU MsgsReplyCB.....	182
4.5.12.12.1	CSU MsgsReplyCB requirements.....	182
4.5.12.12.2	CSU MsgsReplyCB design.....	182
4.5.12.13	CSU MsgsReportCB.....	183
4.5.12.13.1	CSU MsgsReportCB requirements.....	183
4.5.12.13.2	CSU MsgsReportCB design.....	183
4.5.13	Sub-Level CSC msgReadCB.....	184
4.5.13.1	CSU MsgReadReadCB.....	184
4.5.13.1.1	CSU MsgReadReadCB requirements.....	185
4.5.13.1.2	CSU MsgReadReadCB design.....	185

4.5.13.2	CSU MsgReadDeleteCB .....	186
4.5.13.2.1	CSU MsgReadDeleteCB requirements. ....	186
4.5.13.2.2	CSU MsgReadDeleteCB design. ....	186
4.5.13.3	CSU PutupMsgRead .....	187
4.5.13.3.1	CSU PutupMsgRead requirements. ....	187
4.5.13.3.2	CSU PutupMsgRead design. ....	187
4.5.13.4	CSU MsgReadReuseNIncCB. ....	188
4.5.13.4.1	CSU MsgReadReuseNIncCB requirements. ....	188
4.5.13.4.2	CSU MsgReadReuseNIncCB design. ....	188
4.5.13.5	CSU MsgReadReuseCB. ....	189
4.5.13.5.1	CSU MsgReadReuseCB requirements. ....	189
4.5.13.5.2	CSU MsgReadReuseCB design. ....	189
4.5.13.6	CSU MsgReplyCB. ....	190
4.5.13.6.1	CSU MsgReplyCB requirements. ....	190
4.5.13.6.2	CSU MsgReplyCB design. ....	190
4.5.14	Sub-Level CSC mtoTmi_call. ....	191
4.5.14.1	CSU MtoClearNRetCB. ....	192
4.5.14.1.1	CSU MtoClearNRetCB requirements. ....	192
4.5.14.1.2	CSU MtoClearNRetCB design. ....	192
4.5.14.2	CSU MtoSaveNRetCB .....	193
4.5.14.2.1	CSU MtoSaveNRetCB requirements. ....	193
4.5.14.2.2	CSU MtoSaveNRetCB design. ....	193
4.5.14.3	CSU MtoSndRoutCB. ....	194
4.5.14.3.1	CSU MtoSndRoutCB requirements. ....	194
4.5.14.3.2	CSU MtoSndRoutCB design. ....	194
4.5.14.4	CSU MtoSend .....	195
4.5.14.4.1	CSU MtoSend requirements. ....	195
4.5.14.4.2	CSU MtoSend design. ....	195
4.5.14.5	CSU MtoSndUrgCB. ....	196
4.5.14.5.1	CSU MtoSndUrgCB requirements. ....	197
4.5.14.5.2	CSU MtoSndUrgCB design. ....	197
4.5.14.6	CSU ClearMto. ....	198
4.5.14.6.1	CSU ClearMto requirements. ....	198
4.5.14.6.2	CSU ClearMto design. ....	198
4.5.15	Sub-Level CSC mto_call. ....	199
4.5.15.1	CSU mtoad_moreCB .....	199
4.5.15.1.1	CSU mtoad_moreCB requirements. ....	199
4.5.15.1.2	CSU mtoad_moreCB design. ....	199
4.5.15.2	CSU mtoad_rotateCB. ....	200
4.5.15.2.1	CSU mtoad_rotateCB requirements. ....	200
4.5.15.2.2	CSU mtoad_rotateCB design. ....	200
4.5.15.3	CSU PutupMto .....	201
4.5.15.3.1	CSU PutupMto requirements. ....	201
4.5.15.3.2	CSU PutupMto design. ....	202
4.5.16	Sub-Level CSC popupMessage. ....	203
4.5.16.1	CSU CreateMessageBox .....	203

4.5.16.1.1	CSU CreateMessageBox requirements.....	203
4.5.16.1.2	CSU CreateMessageBox design.....	203
4.5.16.2	CSU createMessage.....	204
4.5.16.2.1	CSU createMessage requirements.....	204
4.5.16.2.2	CSU createMessage design.....	204
4.5.16.3	CSU okbutton.....	205
4.5.16.3.1	CSU okbutton requirements.....	205
4.5.16.3.2	CSU okbutton design.....	205
4.5.16.4	CSU unmanageMessage.....	206
4.5.16.4.1	CSU unmanageMessage requirements.....	206
4.5.16.4.2	CSU unmanageMessage design.....	206
4.5.16.5	CSU popupTransientMessage.....	207
4.5.16.5.1	CSU popupTransientMessage requirements.....	207
4.5.16.5.2	CSU popupTransientMessage design.....	207
4.5.16.6	CSU popupMessageBox.....	208
4.5.16.6.1	CSU popupMessageBox requirements.....	208
4.5.16.6.2	CSU popupMessageBox design.....	208
4.5.17	Sub-Level CSC reportCB.....	209
4.5.17.1	CSU ReportMtoCB.....	210
4.5.17.1.1	CSU ReportMtoCB requirements.....	210
4.5.17.1.2	CSU ReportMtoCB design.....	210
4.5.17.2	CSU ReportSplashCB.....	211
4.5.17.2.1	CSU ReportSplashCB requirements.....	211
4.5.17.2.2	CSU ReportSplashCB design.....	211
4.5.17.3	CSU ReportReqCB.....	212
4.5.17.3.1	CSU ReportReqCB requirements.....	212
4.5.17.3.2	CSU ReportReqCB design.....	212
4.5.17.4	CSU ReportMovcmdCB.....	213
4.5.17.4.1	CSU ReportMovcmdCB requirements.....	213
4.5.17.4.2	CSU ReportMovcmdCB design.....	213
4.5.17.5	CSU ReportShotCB.....	214
4.5.17.5.1	CSU ReportShotCB requirements.....	214
4.5.17.5.2	CSU ReportShotCB design.....	214
4.5.17.6	CSU ReportSpotCB.....	215
4.5.17.6.1	CSU ReportSpotCB requirements.....	215
4.5.17.6.2	CSU ReportSpotCB design.....	215
4.5.17.7	CSU ReportFreeTxtCB.....	216
4.5.17.7.1	CSU ReportFreeTxtCB requirements.....	216
4.5.17.7.2	CSU ReportFreeTxtCB design.....	216
4.5.17.8	CSU ReportSaveExitCB.....	217
4.5.17.8.1	CSU ReportSaveExitCB requirements.....	217
4.5.17.8.2	CSU ReportSaveExitCB design.....	217
4.5.17.9	CSU ReportMsgsCB.....	218
4.5.17.9.1	CSU ReportMsgsCB requirements.....	218
4.5.17.9.2	CSU ReportMsgsCB design.....	218
4.5.17.10	CSU PutupReport.....	219



4.5.17.10.1	CSU PutupReport requirements.....	219
4.5.17.10.2	CSU PutupReport design.....	219
4.5.18	Sub-Level CSC reqtTmi_call.....	220
4.5.18.1	CSU ReqClearNRetCB.....	220
4.5.18.1.1	CSU ReqClearNRetCB requirements.....	220
4.5.18.1.2	CSU ReqClearNRetCB design.....	220
4.5.18.2	CSU ReqSaveNRetCB.....	221
4.5.18.2.1	CSU ReqSaveNRetCB requirements.....	221
4.5.18.2.2	CSU ReqSaveNRetCB design.....	222
4.5.18.3	CSU ReqSndRoutCB.....	223
4.5.18.3.1	CSU ReqSndRoutCB requirements.....	223
4.5.18.3.2	CSU ReqSndRoutCB design.....	223
4.5.18.4	CSU ReqSend.....	224
4.5.18.4.1	CSU ReqSend requirements.....	224
4.5.18.4.2	CSU ReqSend design.....	224
4.5.18.5	CSU ReqSndUrgCB.....	225
4.5.18.5.1	CSU ReqSndUrgCB requirements.....	225
4.5.18.5.2	CSU ReqSndUrgCB design.....	225
4.5.18.6	CSU ClearReq.....	226
4.5.18.6.1	CSU ClearReq requirements.....	226
4.5.18.6.2	CSU ClearReq design.....	226
4.5.19	Sub-Level CSC reqt_call.....	227
4.5.19.1	CSU type_moreCB.....	227
4.5.19.1.1	CSU type_moreCB requirements.....	228
4.5.19.1.2	CSU type_moreCB design.....	228
4.5.19.2	CSU type_rotateCB.....	229
4.5.19.2.1	CSU type_rotateCB requirements.....	229
4.5.19.2.2	CSU type_rotateCB design.....	229
4.5.19.3	CSU recon_rotateCB.....	230
4.5.19.3.1	CSU recon_rotateCB requirements.....	230
4.5.19.3.2	CSU recon_rotateCB design.....	230
4.5.19.4	CSU reqtad_moreCB.....	231
4.5.19.4.1	CSU reqtad_moreCB requirements.....	231
4.5.19.4.2	CSU reqtad_moreCB design.....	231
4.5.19.5	CSU reqtad_rotateCB.....	232
4.5.19.5.1	CSU reqtad_rotateCB requirements.....	232
4.5.19.5.2	CSU reqtad_rotateCB design.....	232
4.5.19.6	CSU PutupReq.....	233
4.5.19.6.1	CSU PutupReq requirements.....	233
4.5.19.6.2	CSU PutupReq design.....	233
4.5.20	Sub-Level CSC reuseTmi_call.....	234
4.5.20.1	CSU ReuseClearNRetCB.....	234
4.5.20.1.1	CSU ReuseClearNRetCB requirements.....	235
4.5.20.1.2	CSU ReuseClearNRetCB design.....	235
4.5.20.2	CSU ReuseSendCB.....	236
4.5.20.2.1	CSU ReuseSend requirements.....	236

4.5.20.2.2	CSU ReuseSend design.....	236
4.5.20.3	CSU ClearReuse .....	237
4.5.20.3.1	CSU ClearReuse requirements. ....	237
4.5.20.3.2	CSU ClearReuse design. ....	237
4.5.21	Sub-Level CSC reuse_call. ....	238
4.5.21.1	CSU reusead_moreCB.....	238
4.5.21.1.1	CSU reusead_moreCB requirements.....	238
4.5.21.1.2	CSU reusead_moreCB design.....	238
4.5.21.2	CSU reusead_rotateCB .....	239
4.5.21.2.1	CSU reusead_rotateCB requirements. ....	239
4.5.21.2.2	CSU reusead_rotateCB design. ....	240
4.5.21.3	CSU PutupReuse.....	240
4.5.21.3.1	CSU PutupReuse requirements.....	241
4.5.21.3.2	CSU PutupReuse design.....	241
4.5.22	Sub-Level CSC shotTmi_call. ....	242
4.5.22.1	CSU ShotClearNRetCB.....	242
4.5.22.1.1	CSU ShotClearNRetCB requirements.....	242
4.5.22.1.2	CSU ShotClearNRetCB design.....	242
4.5.22.2	CSU ShotSaveNRetCB .....	243
4.5.22.2.1	CSU ShotSaveNRetCB requirements. ....	243
4.5.22.2.2	CSU ShotSaveNRetCB design. ....	243
4.5.22.3	CSU ShotSndRoutCB.....	244
4.5.22.3.1	CSU ShotSndRoutCB requirements.....	245
4.5.22.3.2	CSU ShotSndRoutCB design.....	245
4.5.22.4	CSU ShotSend .....	246
4.5.22.4.1	CSU ShotSend requirements. ....	246
4.5.22.4.2	CSU ShotSend design. ....	246
4.5.22.5	CSU ShotSndUrgCB.....	247
4.5.22.5.1	CSU ShotSndUrgCB requirements.....	247
4.5.22.5.2	CSU ShotSndUrgCB design.....	247
4.5.22.6	CSU ClearShot.....	248
4.5.22.6.1	CSU ClearShot requirements.....	248
4.5.22.6.2	CSU ClearShot design.....	248
4.5.23	Sub-Level CSC shot_call. ....	249
4.5.23.1	CSU shotad_moreCB.....	249
4.5.23.1.1	CSU shotad_moreCB requirements.....	249
4.5.23.1.2	CSU shotad_moreCB design.....	250
4.5.23.2	CSU shotad_rotateCB .....	250
4.5.23.2.1	CSU shotad_rotateCB requirements. ....	251
4.5.23.2.2	CSU shotad_rotateCB design. ....	251
4.5.23.3	CSU PutupShot .....	252
4.5.23.3.1	CSU PutupShot requirements. ....	252
4.5.23.3.2	CSU PutupShot design. ....	252
4.5.24	Sub-Level CSC splashTmi_call.....	253
4.5.24.1	CSU SplashClearNRetCB.....	253
4.5.24.1.1	CSU SplashClearNRetCB requirements.....	253

4.5.24.1.2	CSU SplashClearNRetCB design.....	253
4.5.24.2	CSU SplashSaveNRetCB.....	254
4.5.24.2.1	CSU SplashSaveNRetCB requirements.....	254
4.5.24.2.2	CSU SplashSaveNRetCB design.....	255
4.5.24.3	CSU SplashSndRoutCB.....	256
4.5.24.3.1	CSU SplashSndRoutCB requirements.....	256
4.5.24.3.2	CSU SplashSndRoutCB design.....	256
4.5.24.4	CSU SplashSend.....	257
4.5.24.4.1	CSU SplashSend requirements.....	257
4.5.24.4.2	CSU SplashSend design.....	257
4.5.24.5	CSU SplashSndUrgCB.....	258
4.5.24.5.1	CSU SplashSndUrgCB requirements.....	258
4.5.24.5.2	CSU SplashSndUrgCB design.....	258
4.5.24.6	CSU ClearSplash.....	259
4.5.24.6.1	CSU ClearSplash requirements.....	259
4.5.24.6.2	CSU ClearSplash design.....	259
4.5.25	Sub-Level CSC splash_call.....	260
4.5.25.1	CSU splashad_moreCB.....	260
4.5.25.1.1	CSU splashad_moreCB requirements.....	261
4.5.25.1.2	CSU splashad_moreCB design.....	261
4.5.25.2	CSU splashad_rotateCB.....	262
4.5.25.2.1	CSU splashad_rotateCB requirements.....	262
4.5.25.2.2	CSU splashad_rotateCB design.....	262
4.5.25.3	CSU PutupSplash.....	263
4.5.25.3.1	CSU PutupSplash requirements.....	263
4.5.25.3.2	CSU PutupSplash design.....	263
4.5.26	Sub-Level CSC spotTmi_call.....	264
4.5.26.1	CSU SpotClearNRetCB.....	264
4.5.26.1.1	CSU SpotClearNRetCB requirements.....	264
4.5.26.1.2	CSU SpotClearNRetCB design.....	264
4.5.26.2	CSU SpotSaveNRetCB.....	265
4.5.26.2.1	CSU SpotSaveNRetCB requirements.....	265
4.5.26.2.2	CSU SpotSaveNRetCB design.....	266
4.5.26.3	CSU SpotSndRoutCB.....	267
4.5.26.3.1	CSU SpotSndRoutCB requirements.....	267
4.5.26.3.2	CSU SpotSndRoutCB design.....	267
4.5.26.4	CSU SpotSend.....	268
4.5.26.4.1	CSU SpotSend requirements.....	268
4.5.26.4.2	CSU SpotSend design.....	268
4.5.26.5.	CSU SpotSndUrgCB.....	269
4.5.26.5.1	CSU SpotSndUrgCB requirements.....	269
4.5.26.5.2	CSU SpotSndUrgCB design.....	269
4.5.26.6	CSU ClearSpot.....	270
4.5.26.6.1	CSU ClearSpot requirements.....	270
4.5.26.6.2	CSU ClearSpot design.....	270
4.5.26.7	CSU KPH_MPHCB.....	272

4.5.26.7.1	CSU KPH_MPHCB requirements.....	272
4.5.26.7.2	CSU KPH_MPHCB design.....	272
4.5.27	Sub-Level CSC spot_call.....	273
4.5.27.1	CSU enemyActivity_moreCB.....	273
4.5.27.1.1	CSU enemyActivity_moreCB requirements.....	273
4.5.27.1.2	CSU enemyActivity_moreCB design.....	273
4.5.27.2	CSU enemyActivity_rotateCB.....	274
4.5.27.2.1	CSU enemyActivity_rotateCB requirements.....	274
4.5.27.2.2	CSU enemyActivity_rotateCB design.....	274
4.5.27.3	CSU enemyType_rotateCB.....	275
4.5.27.3.1	CSU enemyType_rotateCB requirements.....	275
4.5.27.3.2	CSU enemyType_rotateCB design.....	276
4.5.27.4	CSU obsInt_rotateCB.....	276
4.5.27.4.1	CSU obsInt_rotateCB requirements.....	277
4.5.27.4.2	CSU obsInt_rotateCB design.....	277
4.5.27.5	CSU spotad_moreCB.....	278
4.5.27.5.1	CSU spotad_moreCB requirements.....	278
4.5.27.5.2	CSU spotad_moreCB design.....	278
4.5.27.6	CSU spotad_rotateCB.....	279
4.5.27.6.1	CSU spotad_rotateCB requirements.....	279
4.5.27.6.2	CSU spotad_rotateCB design.....	279
4.5.27.7	CSU direc_rotateCB.....	280
4.5.27.7.1	CSU direc_rotateCB requirements.....	280
4.5.27.7.2	CSU direc_rotateCB design.....	280
4.5.27.8	CSU PutupSpot.....	281
4.5.27.8.1	CSU PutupSpot requirements.....	281
4.5.27.8.2	CSU PutupSpot design.....	281
4.5.28	Sub-Level CSC sysMainCB.....	282
4.5.28.1	CSU SysMainMsgsCB.....	282
4.5.28.1.1	CSU SysMainMsgsCB requirements.....	282
4.5.28.1.2	CSU SysMainMsgsCB design.....	282
4.5.28.2	CSU SysMainRprtCB.....	283
4.5.28.2.1	CSU SysMainRprtCB requirements.....	283
4.5.28.2.2	CSU SysMainRprtCB design.....	284
4.5.28.3	CSU SysMainAddListCB.....	284
4.5.28.3.1	CSU SysMainAddListCB requirements.....	285
4.5.28.3.2	CSU SysMainAddListCB design.....	285
4.5.28.4	CSU SysMainGrpListCB.....	286
4.5.28.4.1	CSU SysMainGrpListCB requirements.....	286
4.5.28.4.2	CSU SysMainGrpListCB design.....	286
4.5.28.5	CSU SysMainLocListCB.....	287
4.5.28.5.1	CSU SysMainLocListCB requirements.....	287
4.5.28.5.2	CSU SysMainLocListCB design.....	287
4.5.28.6	CSU SysMainLogoutCB.....	288
4.5.28.6.1	CSU SysMainLogoutCB requirements.....	288
4.5.28.6.2	CSU SysMainLogoutCB design.....	288

4.5.28.7	CSU PutupSysMain.....	289
4.5.28.7.1	CSU PutupSysMain requirements.....	289
4.5.28.7.2	CSU PutupSysMain design.....	289
4.5.29	Sub-Level CSC util.....	290
4.5.29.1	CSU init_selections.....	290
4.5.29.1.1	CSU init_selections requirements.....	290
4.5.29.1.2	CSU init_selections design.....	290
4.5.29.2	CSU rotate.....	291
4.5.29.2.1	CSU rotate requirements.....	291
4.5.29.2.2	CSU rotate design.....	291
4.5.29.3	CSU moreAddress .....	292
4.5.29.3.1	CSU moreAddress requirements.....	292
4.5.29.3.2	CSU moreAddress design.....	292
4.5.29.4	CSU moreLctn.....	293
4.5.29.4.1	CSU moreLctn requirements.....	293
4.5.29.4.2	CSU moreLctn design.....	293
4.5.29.5	CSU string2utm.....	294
4.5.29.5.1	CSU string2utm requirements.....	294
4.5.29.5.2	CSU string2utm design.....	295
4.5.29.6	CSU UpdateEnvelope .....	295
4.5.29.6.1	CSU UpdateEnvelope requirements.....	295
4.5.29.6.2	CSU UpdateEnvelope design.....	296
4.5.30	Sub-Level CSC Builder Xcessory Functions.....	297
4.5.30.1.	CSU CreateAddrList .....	297
4.5.30.2.	CSU Createform requirements.....	297
4.5.30.3.	CSU formTrim .....	297
4.5.30.4.	CSU CreateFreeTxtForm .....	297
4.5.30.5.	CSU CreateGrpListForm.....	297
4.5.30.6.	CSU CreateLocListForm .....	297
4.5.30.7	CSU CreateLogonForm.....	297
4.5.30.8	CSU CreatemovcmdTmi.....	297
4.5.30.9	CSU CreatemovcmdForm .....	297
4.5.30.10	CSU CreateMsg1Form.....	297
4.5.30.11	CSU CreateMsgReadForm .....	298
4.5.30.12	CSU CreatemtoTmi.....	298
4.5.30.13	CSU CreatemtoForm.....	298
4.5.30.14	CSU CreateReportForm.....	298
4.5.30.15	CSU CreatereqtTmi.....	298
4.5.30.16	CSU CreatereqtForm .....	298
4.5.30.17	CSU CreatereuseTmi.....	298
4.5.30.18	CSU CreatereuseForm .....	298
4.5.30.19	CSU CreateshotTmi.....	298
4.5.30.20	CSU CreateshotForm .....	298
4.5.30.21	CSU CreatesplashTmi.....	299
4.5.30.22	CSU CreatesplashForm.....	299
4.5.30.23.	CSU CreatespotTmi.....	299

4.5.30.24	CSU CreatespotForm.....	299
4.5.30.25	CSU CreateSysMainForm.....	299
4.6	CSC pdu decode.....	299
4.6.1	CSU pdu_decode.....	300
4.6.1.1	CSU pdu_decode requirements.....	300
4.6.1.2	CSU pdu_decode design.....	301
4.6.2	CSU pdu_spec_decode.....	305
4.6.2.1	CSU pdu_spec_decode requirements.....	306
4.6.2.2	CSU pdu_spec_decode design.....	306
4.6.3	CSU decode_ack_pdu.....	308
4.6.3.1	CSU decode_ack_pdu requirements.....	308
4.6.3.2	CSU decode_ack_pdu design.....	308
4.6.4	CSU decode_spot_pdu.....	310
4.6.4.1	CSU decode_spot_pdu requirements.....	310
4.6.4.2	CSU decode_spot_pdu design.....	310
4.6.5	CSU decode_bda_pdu.....	313
4.6.5.1	CSU decode_bda_pdu requirements.....	314
4.6.5.2	CSU decode_bda_pdu design.....	314
4.6.6	CSU decode_free_text_pdu.....	316
4.6.6.1	CSU decode_free_text_pdu requirements.....	316
4.6.6.2	CSU decode_free_text_pdu design.....	316
4.6.7	CSU decode_gndroute_pdu.....	317
4.6.7.1	CSU decode_gndroute_pdu requirements.....	318
4.6.7.2	CSU decode_gndroute_pdu design.....	318
4.6.8	CSU decode_airroute_pdu.....	320
4.6.8.1	CSU decode_airroute_pdu requirements.....	320
4.6.8.2	CSU decode_airroute_pdu design.....	320
4.6.9	CSU decode_bridge_pdu.....	322
4.6.9.1	CSU decode_bridge_pdu requirements.....	322
4.6.9.2	CSU decode_bridge_pdu design.....	322
4.6.10	CSU decode_lzpz_pdu.....	325
4.6.10.1	CSU decode_lzpz_pdu requirements.....	325
4.6.10.2	CSU decode_lzpz_pdu design.....	325
4.6.11	CSU decode_bpop_pdu.....	328
4.6.11.1	CSU decode_bpop_pdu requirements.....	328
4.6.11.2	CSU decode_bpop_pdu design.....	328
4.6.12	CSU decode_crossing_pdu.....	330
4.6.12.1	CSU decode_crossing_pdu requirements.....	330
4.6.12.2	CSU decode_crossing_pdu design.....	330
4.6.14	CSU decode_repeat_pdu.....	332
4.6.14.1	CSU decode_repeat_pdu requirements.....	332
4.6.14.2	CSU decode_repeat_pdu design.....	332
4.6.15	CSU decode_pirep_pdu.....	334
4.6.15.1	CSU decode_pirep_pdu requirements.....	334
4.6.15.2	CSU decode_pirep_pdu design.....	334
4.6.16	CSU decode_cancel_pdu.....	338

4.6.16.1	CSU decode_cancel_pdu requirements.....	338
4.6.16.2	CSU decode_cancel_pdu design.....	338
4.6.17	CSU decode_check_pdu.....	340
4.6.17.1	CSU decode_check_pdu requirements.....	340
4.6.17.2	CSU decode_check_pdu design.....	340
4.6.18	CSU decode_cno_pdu.....	341
4.6.18.1	CSU decode_cno_pdu requirements.....	341
4.6.18.2	CSU decode_cno_pdu design.....	342
4.6.19	CSU decode_shift_pdu.....	343
4.6.19.1	CSU decode_shift_pdu requirements.....	344
4.6.19.2	CSU decode_shift_pdu design.....	344
4.6.20	CSU decode_nwmsn_pdu.....	346
4.6.20.1	CSU decode_nwmsn_pdu requirements.....	346
4.6.20.2	CSU decode_nwmsn_pdu design.....	346
4.6.21	CSU decode_mto_pdu.....	349
4.6.21.1	CSU decode_mto_pdu requirements.....	350
4.6.21.2	CSU decode_mto_pdu design.....	350
4.6.22	CSU decode_shot_pdu.....	352
4.6.22.1	CSU decode_shot_pdu requirements.....	353
4.6.22.2	CSU decode_shot_pdu design.....	353
4.6.23	CSU decode_splash_pdu.....	355
4.6.23.1	CSU decode_splash_pdu requirements.....	355
4.6.23.2	CSU decode_splash_pdu design.....	355
4.6.24	CSU decode_eom_pdu.....	357
4.6.24.1	CSU decode_eom_pdu requirements.....	357
4.6.24.2	CSU decode_eom_pdu design.....	357
4.6.25	CSU decode_status_pdu.....	360
4.6.25.1	CSU decode_status_pdu requirements.....	360
4.6.25.2	CSU decode_status_pdu design.....	360
4.6.26	CSU decode_request_pdu.....	363
4.6.26.1	CSU decode_request_pdu requirements.....	363
4.6.26.2	CSU decode_request_pdu design.....	363
4.6.27	CSU decode_nbc1_pdu.....	365
4.6.27.1	CSU decode_nbc1_pdu requirements.....	365
4.6.27.2	CSU decode_nbc1_pdu design.....	365
4.6.28	CSU decode_nbc4_pdu.....	368
4.6.28.1	CSU decode_nbc4_pdu requirements.....	368
4.6.28.2	CSU decode_nbc4_pdu design.....	368
4.6.29	CSU decode_nbc5_pdu.....	371
4.6.29.1	CSU decode_nbc5_pdu requirements.....	371
4.6.29.2	CSU decode_nbc5_pdu design.....	371
4.6.30	CSU decode_miji_pdu.....	373
4.6.30.1	CSU decode_miji_pdu requirements.....	373
4.6.30.2	CSU decode_miji_pdu design.....	373
4.6.31	CSU decode_dnav_pdu.....	376
4.6.31.1	CSU decode_dnav_pdu requirements.....	376

4.6.31.2	CSU decode_dnav_pdu design.....	376
4.6.32	CSU decode_movcmd_pdu.....	378
4.6.32.1	CSU decode_movcmd_pdu requirements.....	378
4.6.32.2	CSU decode_movcmd_pdu design.....	378
4.6.33	CSU puttogether.....	380
4.6.33.1	CSU puttogether requirements.....	381
4.6.33.2	CSU puttogether design.....	381
4.6.34	CSU decode_time.....	382
4.6.34.1	CSU decode_time requirements.....	382
4.6.34.2	CSU decode_time design.....	382
4.6.35	CSU freq_hertz.....	383
4.6.35.1	CSU freq_hertz requirements.....	384
4.6.35.2	CSU freq_hertz design.....	384
4.6.36	CSU sprintf_UTM.....	385
4.6.36.1	CSU sprintf_UTM requirements.....	385
4.6.36.2	CSU sprintf_UTM design.....	385
4.6.37	CSU sprintf_speed.....	386
4.6.37.1	CSU sprintf_speed requirements.....	386
4.6.37.2	CSU sprintf_speed design.....	387
5.	DMCC Global Data Elements.....	389
6.	CSCI data files.....	471
7.	Requirements traceability.....	472
8.	Notes.....	472
10.	Appendix A - DMCC Software Structure Charts.....	473
10.1	DMCC CSCI.....	474
10.2	CSC dms.....	475
10.3	CSC build_pdu.....	476
10.4	CSC ipc.....	477
10.5	CSC dmcc_sim_tx.....	478
10.6	Sublevel CSC dmcc_sim_rx.....	479
10.7	CSC pdu_decode.....	480
10.8	CSC client.....	481
20.	Appendix B - DMCC "C" Source Data Structures.....	482
30.	Appendix C - DMCC Client State Transition Diagram.....	483
40.	Appendix D - DMCC Protocol Data Units (PDUs).....	510
40.1	Simnet Header and common block.....	511
40.2	DIS Header and Common Block.....	512
40.3	BDA PDU Specific.....	513
40.4	MOVCMD PDU Specific.....	514
40.5	MIJI PDU Specific.....	515
40.6	PIREP PDU Specific.....	516
40.7	RECON GND ROUTE PDU Specific:.....	517
40.8	RECON AIR ROUTE Specific:.....	518
40.9	RECON Bridge Specific:.....	519
40.10	RECON LZ/PZ PDU Specific.....	520
40.11	RECON BP-OP PDU Specific:.....	521



40.12	RECON Crossing PDU Specific: .....	522
40.13	ACK PDU Specific.....	523
40.14	SPOT PDU Specific.....	524
40.15	NBC-1 PDU Specific.....	525
40.16	NBC-4 PDU Specific.....	526
40.17	NBC-5 PDU Specific.....	527
40.18	ARTILLERY REPEAT PDU Specific.....	528
40.19	ARTILLERY CANCEL PDU Specific.....	528
40.20	ARTILLERY CHECK PDU Specific.....	528
40.21	ARTILLERY CNO PDU Specific.....	528
40.22	ARTILLERY SHIFT PDU Specific.....	529
40.23	ARTILLERY SHOT PDU Specific.....	529
40.24	ARTILLERY SPLASH PDU Specific.....	530
40.25	ARTILLERY EOM PDU Specific.....	530
40.26	STATUS PDU Specific.....	531
40.27	REQUEST PDU Specific.....	532
40.28	DNAV PDU Specific.....	533
40.29	FREE TEXT PDU Specific.....	534
50.	Appendix E - DMCC Functional Development Tests.....	535
50.1.	Single Client Stand Alone IPC Thread Test.....	536
50.2.	Dual Client Stand Alone IPC Thread Test.....	538
50.3.	Transmitted PDU Content Accuracy Test.....	541
50.4.	DMCC PDU Reception Accuracy Test.....	543
50.5.	DMCC X-terminal Configuration Test.....	546
50.6.	Dual Host Dual Client SIMNET Compliance Test.....	548
50.7.	Window Content Verification Test.....	549
50.8.	Graphical User Interface Window Navigation Test.....	552
50.9.	Re-use and Reply .....	557
50.10.	Miscellaneous Functional Tests .....	560
50.11.	Functional Stress Tests .....	560
50.12	Functional Test... Window Widgets.....	561
60.	Appendix F - DMCC Graphical User Interface.....	565
60.1	Purpose and Scope:.....	565
60.2	General Description of Operation: .....	566
60.2.1	Interface Elements .....	566
60.2.2	Menu Windows.....	566
60.3	DMCC Incoming Message Control.....	567
60.3.1	Overview.....	567
60.3.2	Call Sign.....	567
60.3.3	Groups.....	567
60.3.4	Addresses.....	567
60.3.5	Reception.....	567
60.3.6	TOC - to - TOC Communications.....	567
60.3.7	Message Reception.....	568
60.3.8	Sender ID & Time Stamp.....	568
60.3.9	Acknowledge .....	568

60.3.10	The Alert Box.....	568
60.3.11	Message Forwarding (RE_USE):.....	568
60.3.12	Reply.....	569
60.4	DMCC Outgoing Message Control (REPORTS).....	570
60.4.1	Available Reports.....	570
60.4.2	Reports Function.....	570
60.4.3	Accessing the Reports Window.....	570
60.4.4	Menu Wundows.....	570
60.4.5	Individual Reports Windows.....	570
60.4.6	CIK MENU'.....	571
60.4.7	LOGON.....	571
60.4.7.1	LOGON TO NETWORK.....	571
60.4.7.2	STAND ALONE OPERATION.....	571
60.4.7.3	SYS MAIN Window.....	572
60.4.8	ADDRESS LIST EDIT.....	572
60.4.8.1	Call Signs, Group Names.....	572
60.4.8.2	Initial Members.....	572
60.4.8.3	ADDRESS LIST.....	572
60.4.8.4	Adding and Deleting Addresses.....	572
60.4.8.5	ADDRESS TEXT WIDGET.....	572
60.4.8.6	SIZING.....	573
60.4.8.7	UP BUTTON.....	573
60.4.8.8	DOWN BUTTON.....	573
60.4.8.9	ADD BUTTON.....	573
60.4.8.10	DEL BUTTON.....	573
60.4.9	CEOI/GROUP LIST EDIT.....	573
60.4.9.1	MESSAGE RECEPTION.....	574
60.4.9.2	EDITING THE LIST.....	574
60.4.9.3	FUNCTIONAL DESCRIPTION.....	574
60.4.10	LOCATION LIST EDIT.....	574
60.4.10.1	FUNCTIONAL DESCRIPTION.....	574
60.4.11	SYS MAIN.....	574
60.4.11.1	NEW MESSAGE Indicator.....	574
60.4.11.2	SYS MAIN TMI BUTTONS.....	575
60.5	SYS MAIN MSG1 Window.....	576
60.5.1	THE IN-BOX.....	576
60.5.2	ENVELOPE ICON.....	576
60.5.3	MSG READ.....	576
60.5.4	Message Display Formats.....	576
60.5.5	PRE MSG and NEXT MSG Bezels.....	577
60.5.6	DELETE MSG TMI.....	577
60.5.7	SAVE & EXIT.....	577
60.5.8	RE-USE.....	577
60.5.9	REPLY.....	577
60.5.10	PREV, NEXT.....	578
60.6	RPRT Windows.....	579

60.6.1	SAVE & EXIT.....	579
60.6.2	SHOT.....	579
60.6.3	SPLASH.....	579
60.6.4	MTO.....	579
60.6.5	SPOT.....	579
60.6.6	MOVCMD.....	580
60.6.7	REQT.....	580
60.6.8	FREE TXT.....	580
60.7	RPRT Window Functionality.....	581
60.7.1	COMMON RPRTS WINDOW FUNCTIONs.....	581
60.7.1.1	ADRS.....	581
60.7.1.2	TAPE WIDGET.....	581
60.7.1.3	TEXT MSG.....	581
60.7.1.4	SEND ROUTIN.....	581
60.7.1.5	SEND URGENT.....	582
60.7.1.6	CLR & RETURN.....	582
60.7.1.7	SAVE & RETURN.....	582
60.7.1.8	ENUMERATED TYPES DEFAULT VALUE.....	582
60.7.2	SPOT REPORT.....	582
60.7.2.1	Enemy Speed.....	583
60.7.2.2	Text Msg TMI.....	583
60.7.2.3	MPH/KPH Switch.....	583
60.7.2.4	Enemy Type.....	583
60.7.2.5	Enemy Number.....	583
60.7.2.6	Enemy Location.....	583
60.7.3	MTO.....	583
60.7.3.1	REQ ADJ.....	584
60.7.3.2	EAT.....	584
60.7.3.3	EOM.....	584
60.7.3.4	TEXT MSG.....	584
60.7.4	SHOT.....	584
60.7.5	SPLASH.....	584
60.7.6	MOVCMD.....	585
60.7.6.1	TASK.....	585
60.7.6.2	When.....	585
60.7.6.3	Location.....	586
60.7.7	REQT REPORT.....	586
60.7.7.1	Report Type Selection.....	586
60.7.7.2	Recon Type Selection.....	586
60.7.8	FREE TEXT.....	587
60.8	Message Display Formats.....	588
70.	Appendix G - Data Flow Diagrams.....	618
70.1	Top Level DMCC Inter-Process Data Flow Diagram.....	618
70.2	DMCC X-Client Process Data Flow Diagram.....	618

## LIST of FIGURES

Figure 1	DMCS Software Architecture.....	2
Figure 2	DMCC X-windows Client Server Environment.....	4
Figure 3	Top Level DMCC CSCs .....	9
Figure 4	DMCC Concurrent Processes .....	21
Figure 5	CSC dms State Transition Diagram .....	22
Figure 6	Client Child State Transition Diagram .....	25
Figure 6a	CSC dms Structure.....	31
Figure 7	build_pdu Structure.....	41
Figure 8	CSC ipc Structure .....	54
Figure 9	Sublevel CSU dmcc_sim_rx Structure.....	71
Figure 10	Sublevel CSU dmcc_sim_tx Structure .....	101
Figure 11	CSC Client Structure Chart.....	114
Figure 12	CSU pdu_decode Structure .....	300
Figure 13	Top Level DMCC CSCI Structure .....	474
Figure 14	CSC dms Structure.....	475
Figure 15	CSC build_pdu Structure.....	476
Figure 16	CSC ipc Structure .....	477
Figure 17	Sublevel CSC dmcc_sim_tx Structure.....	478
Figure 18	Sublevel CSC dmcc_sim_rx Structure.....	479
Figure 19	CSC pdu_decode Structure.....	480
Figure 20	CSC client Structure .....	481
Figure 21	Simnet PDU Header & Common.....	511
Figure 22	Simnet PDU Header & Common.....	512
Figure 23	Battle Damage PDU Specific.....	513
Figure 24	MOVCMD PDU Specific.....	514
Figure 25	MIJI PDU Specific.....	515
Figure 26	PIREP PDU Specific.....	516
Figure 27	Gnd Route Recon PDU Specific.....	517
Figure 28	Air Route Recon PDU Specific.....	518
Figure 29	Bridge Recon PDU Specific .....	519
Figure 30	LZ/PZ Recon PDU Specific .....	520
Figure 31	BP-OP Recon PDU Specific.....	521
Figure 32	Crossing Recon PDU Specific .....	522
Figure 33	Acknowledge PDU Specific.....	523
Figure 34	Acknowledge PDU Specific.....	524
Figure 35	NBC-1 PDU Specific.....	525
Figure 36	NBC-4 PDU Specific.....	526
Figure 37	NBC-5 PDU Specific.....	527
Figure 38	Artillery Repeat PDU Specific .....	528
Figure 39	Artillery Cancel PDU Specific.....	528
Figure 40	Artillery Check PDU Specific.....	528
Figure 41	Artillery CNO PDU Specific.....	528
Figure 42	Artillery Shift PDU Specific.....	529

Figure 43	Artillery Shot PDU Specific .....	529
Figure 44	Artillery Splash PDU Specific.....	530
Figure 45	Artillery EOM PDU Specific.....	530
Figure 46	Artillery Status PDU Specific.....	531
Figure 47	Artillery Request PDU Specific .....	532
Figure 48	DNAV PDU Specific.....	533
Figure 49	Free Text PDU Specific.....	534

## LIST of TABLES

Table 1.	CSC Summary table.....	10
Table 2	Memory Requirements .....	19
Table 3.	CSC/CSU Cross Reference Table. ....	30
Table 4	Structure assocPDU_Type.....	390
Table 5	assocPDUWithLLCHdrType .....	390
Table 6	assocPDU structure ActivateRequestEntry.....	390
Table 7	assocPDU structure ActivateRequestBufferStructure.....	390
Table 8	assoc_PDU.h define statements.....	391
Table 9	structure InderectFireDataBase_type.....	392
Table 10	structure DetonationDataBase_type.....	392
Table 11	structure VehicleAppearanceDataBase_type.....	392
Table 12	structure StateDataBase_type.....	393
Table 13	structure SIMActivateRequestDataBase_type.....	393
Table 14	databases.h define statements .....	394
Table 15	declare_netif dimension statements.....	395
Table 16	declar_vars structures.....	396
Table 17	declar_vars dimension statements.....	396
Table 18	decode_defines structure Non_specific.....	400
Table 19	decode_defines define statements .....	401
Table 20	decode_list dimension statements .....	403
Table 21	decode_strs dimension statements.....	404
Table 22	DIS_PDU structure Header_type .....	406
Table 23	DIS_PDU structure ActivateRequest_type .....	406
Table 24	DIS_PDU structure ActivateResponse_type .....	407
Table 25	DIS_PDU structure DectivateRequest_type.....	407
Table 26	DIS_PDU structure DeactivateResposnse_type.....	407
Table 27	DIS_PDU structure Collision_type.....	408
Table 28	DIS_PDU structure Collision_type.....	408
Table 29	DIS_PDU structure Emitter_Type .....	409
Table 30	DIS_PDU structure EntityState_type.....	409
Table 31	DIS_PDU structure Fire_type .....	410
Table 32	DIS_PDU structure Radar_type.....	410
Table 33	DIS_PDU RepairComplete_type.....	410
Table 34	DIS_PDU structure ReparirResponse_type.....	411
Table 35	DIS_PDU structure Resupply_type .....	411
Table 36	DIS_PDU structure ResupplyCancel_type.....	411
Table 37	DIS_PDU structure ServiceRequest_type .....	412
Table 38	DIS_PDU structure UpdateThresholdRequest_type ...	412
Table 39	DIS_PDU structure UpdateThresholdResponse_type .....	412
Table 40	DIS_types structure Land.....	413
Table 41	DIS_types structure Land.....	413
Table 42	DIS_types structure Surface.....	414
Table 43	DIS_types structure Subsurface .....	414
Table 44	DIS_types structure Space.....	414

Table 45	DIS_types structure GuidedMunitions.....	415
Table 46	DIS_types structure Lifeforms .....	415
Table 47	DIS_types structure Environmentals.....	415
Table 48	DIS_types structure CulturalFeatures.....	415
Table 49	DIS_types structure ID_type.....	416
Table 50	DIS_types structure Event_ID_type.....	416
Table 51	DIS_types structure Event_ID_type.....	416
Table 52	DIS_types structure Supply_type.....	417
Table 53	DIS_types structure DefaultThresholds_type.....	417
Table 54	DIS_types structure WorldCoord_type.....	417
Table 55	DIS_types structure EntityCoord_type.....	417
Table 56	DIS_types structure AngularVelocity_type.....	418
Table 57	DIS_types structure FloatAngles_type.....	418
Table 58	DIS_types structure LinearVelocity_type.....	418
Table 59	DIS_types structure Orientation_type.....	418
Table 60	DIS_types structure DeadReconing_type.....	419
Table 61	DIS_types structure EntityMarking_type.....	419
Table 62	DIS_types structure TerrainDBaseID_type.....	419
Table 63	DIS_types structure BurstDescription_type.....	420
Table 64	DIS_types structure Army .....	420
Table 65	DIS_types structure AirForce.....	420
Table 66	DIS_types structure CoastGuard.....	421
Table 67	DIS_types structure Marines.....	421
Table 68	DIS_types structure Navy.....	421
Table 69	DIS_types structure Unit_type.....	422
Table 70	DIS_types structure EntityCapabilities_type.....	422
Table 71	DIS_types structure RadarSystems_type.....	422
Table 72	DIS_types structure Illumined_type.....	423
Table 73	DIS_types structure Sweep_type.....	423
Table 74	DIS_types structure Emitters_Type_type.....	423
Table 75	DIS_types structure Emitters_Data_type.....	423
Table 76	DIS_types structure Emitters_type.....	424
Table 77	DIS_types structure RadarBaseDate_type.....	424
Table 78	DIS_types structure RadarArray_type.....	424
Table 79	DIS_types structure TimeStamp_type.....	425
Table 80	DIS_types define statements .....	425
Table 81	DIS_types structure DMC_CommonBlock_type.....	436
Table 82	DIS_types structure Header_and_commonBlock_type.....	437
Table 83	DIS_types structure ACK_Specific_type.....	438
Table 84	DIS_types structure ACK_type.....	438
Table 85	DIS_types structure DISAck_type.....	438
Table 86	DIS_types structure Spot_Specific_type.....	439
Table 87	DIS_types structure Spot_type.....	439
Table 88	DIS_types structure DISSpot_type.....	439
Table 89	DIS_types structure BDA_Specific_type.....	440
Table 90	DIS_types structure BDA_type.....	440

Table 91	DIS_types structure DISBDA_type .....	440
Table 92	DIS_types structure GndRoute.....	441
Table 93	DIS_types structure AirRoute.....	441
Table 94	DIS_types structure Bridge .....	441
Table 95	DIS_types structure LZ_PZ.....	442
Table 96	DIS_types structure BP_OP .....	442
Table 97	DIS_types structure Crossing .....	442
Table 98	DIS_types structure Recon_Specific_type.....	443
Table 99	DIS_types structure Recon_type .....	443
Table 100	DIS_types structure DisRecon_type .....	443
Table 101	DIS_types structure Repeat.....	443
Table 102	DIS_types structure Cancel.....	444
Table 103	DIS_types structure CheckFire.....	444
Table 104	DIS_types structure CNO .....	444
Table 105	DIS_types structure Shift .....	444
Table 106	DIS_types structure NewMission .....	445
Table 107	DIS_types structure MTO.....	445
Table 108	DIS_types structure Shot.....	445
Table 109	DIS_types structure Splash.....	446
Table 110	DIS_types structure EndOfMission.....	446
Table 111	DIS_types structure Artillery_Specific_type .....	446
Table 112	DIS_types structure Artillery_type.....	447
Table 113	DIS_types structure DISArtillery_type .....	447
Table 114	DIS_types structure Move_Specific_type .....	447
Table 115	DIS_types structure Move_type.....	448
Table 116	DIS_types structure DISMove_type .....	448
Table 117	DIS_types structure Status_Specific_type.....	448
Table 118	DIS_types structure Status_type.....	449
Table 119	DIS_types structure DISStatus_type.....	449
Table 120	DIS_types structure Request_Specific_type.....	449
Table 121	DIS_types structure Request_type .....	449
Table 122	DIS_types structure DISRequest_type.....	450
Table 123	DIS_types structure DMC_CommonBlock_type .....	450
Table 124	DIS_types structure NBC_4.....	450
Table 125	DIS_types structure DMC_CommonBlock_type .....	451
Table 126	DIS_types structure NBC_Specific_type .....	451
Table 127	DIS_types structure NBC_type.....	451
Table 128	DIS_types structure DISNBC_type .....	451
Table 129	DIS_types structure MIJI_Specific_type .....	452
Table 130	DIS_types structure MIJI_type.....	452
Table 131	DIS_types structure DISMIJI_type .....	452
Table 132	DIS_types structure PIREP_Specific_type.....	453
Table 133	DIS_types structure PIREP_type.....	453
Table 134	DIS_types structure DISPIREP_type.....	453
Table 135	DIS_types structure DNAV_Specific_type.....	454
Table 136	DIS_types structure DNAV_type.....	454



Table 137	DIS_types structure DISDNAV_type.....	454
Table 138	DIS_types structure FreeText_Specific_type .....	454
Table 139	DIS_types structure FreeText_type.....	455
Table 140	DIS_types structure DISFreeText_type .....	455
Table 141	DIS_types structure DMCPDU_type.....	455
Table 142	DIS_types structure DMCDISPDU_type.....	456
Table 143	DIS_types define statements .....	456
Table 144	DMC_types structure DTG_type .....	458
Table 145	DMC_types structure EncodingScheme_type.....	458
Table 146	DMC_types structure UTM_type.....	458
Table 147	DMC_types structure Speed_type.....	459
Table 148	DMC_types define statements .....	459
Table 149	dms structure msgdests_t.....	465
Table 150	dms structure dms_shmem_t.....	465
Table 151	dms structure dms_outgoingmsg_t.....	465
Table 152	dms structure log .....	466
Table 153	dms structure dms_msg_t.....	466
Table 154	dms define statements.....	466
Table 155	global_netif define statements.....	468
Table 156	global_vars define statements .....	469
Table 157	dms structure Int_Array .....	469
Table 158	netif define statements .....	470
Table 159	DMCC CSC/CSU Data File Map.....	470
Table 160	Client State Transition Table.....	484

## **1. Scope.**

### **1.1 Identification.**

**Airnet Digital Message Communications Console CSCI.**

### **1.2 System overview.**

The AIRNET system is a network of simulators to help to teach Army aviation crews and battlefield support personnel to fight in a combined arms battlefield environment.

The AIRNET Digital Message Communications Console (DMCC) provides a capability for pre-formatted and free text digital messaging between simulation entities which are internal and external to the AIRNET network.

The DMCC software is a multi-threaded, X-windows client/server application suite.

The DMCC software contains a custom protocol for communications in the context of the SIMNET and Draft DIS 2.0 distributed simulation protocols. This protocol is fully interoperable across the SIMNET/DIS 2.0 Protocol Translator Gateway.

The DMCC uses a graphical user interface to emulate vehicle crew station soldier machine interfaces.

The Digital Message Communications Console allows transmission, reception, and storage of, and access to, pre-formatted and free text tactical messages between ground support, Tactical Operations Centers, Fire Support Elements, and manned vehicle simulators, via the SIMNET and DIS simulation networks.

The DMCC software is platformed on a Sun Microsystems Sparc-10 workstation running SunOS, Sun Microsystems Software's implementation of the UNIX operating system. The runtime environment includes Version 11 of the X-Windows windowing system and OSF Motif version 1.1 graphical user interface.

Up to 8 Wyse X-terminals may be connected to the workstation via the DMCC dedicated Ethernet local area network. Multiple DMCC client executables may be operational concurrently under the X client/server model.

DMCC client processes run on the Sun hardware. "Local client" operation is not used in this implementation.

The following diagram shows the top-level architecture of the Digital Message Communications Console System.

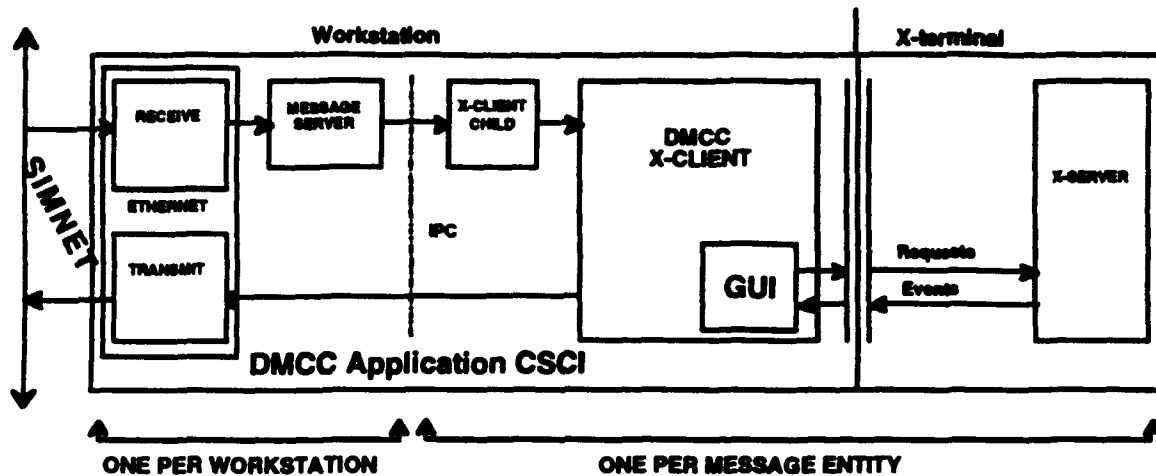


Figure 1 DMCS Software Architecture

The Receive process monitors the Ethernet for incoming messages and informs the Message Server of their arrival.

The Message Server checks the message to see if it is addressed to any of the resident DMCC X-client applications. If the message is addressed to one or more of the DMCC X-client applications, then the Message Server places the message in Shared Memory and informs the affected DMCC X-client applications of its location through a message queue which is monitored by the Client Child processes.

The Client Child processes are spawned at logon by the X-Client processes and wait for indication of incoming messages addressed to its parent. The Client Child processes then inform the Client of the message arrival through a pipe.

The DMCC X-Client process is instantiated for each DMCC entity active in the workstation. The DMCC X-Client process contains the message queue, user interface, and message preparation and dispatch software. The X-Client also contains the software which allows the user to set up addresses to which the user wishes to transmit messages, groups to which the user wishes to receive messages, and locations which are referred to in creating messages for transmission.

The DMCC X-Client software has windows for six different purposes, including:

- (1) Logon and Setup
- (2) Message Queue Access
- (3) Message Access/Deletion
- (4) Message Retrieval
- (5) Message Forwarding
- (6) Report Preparation and Transmission

Each instantiation of the DMCC maintains a list of people or teams to whom it can send messages called the **Address List**. The user sets up the list of addresses prior to the start of the exercise, in order to quickly send messages during the exercise. He may change the list during the exercise, but normally he would not want to do this.

For example, the user may want to be able to send messages during the exercise to his battalion commander, his wing man, the Fire Support Element, etc. The OPORDS for his mission will include this list. He will probably want to enter their CEOs (call signs) in the Address List.

Groups are collections of DMCC simulation entities. For example, several entities may be members of the same team, say TEAM A. If they wish to receive messages transmitted to TEAM A, they must tell their DMCCs to receive messages addressed to that group. Each DMCC allows its operator to be a member of up to seven groups.

The following figure details the nature of the instantiated-client configuration in the context of a X-terminal client server environment:

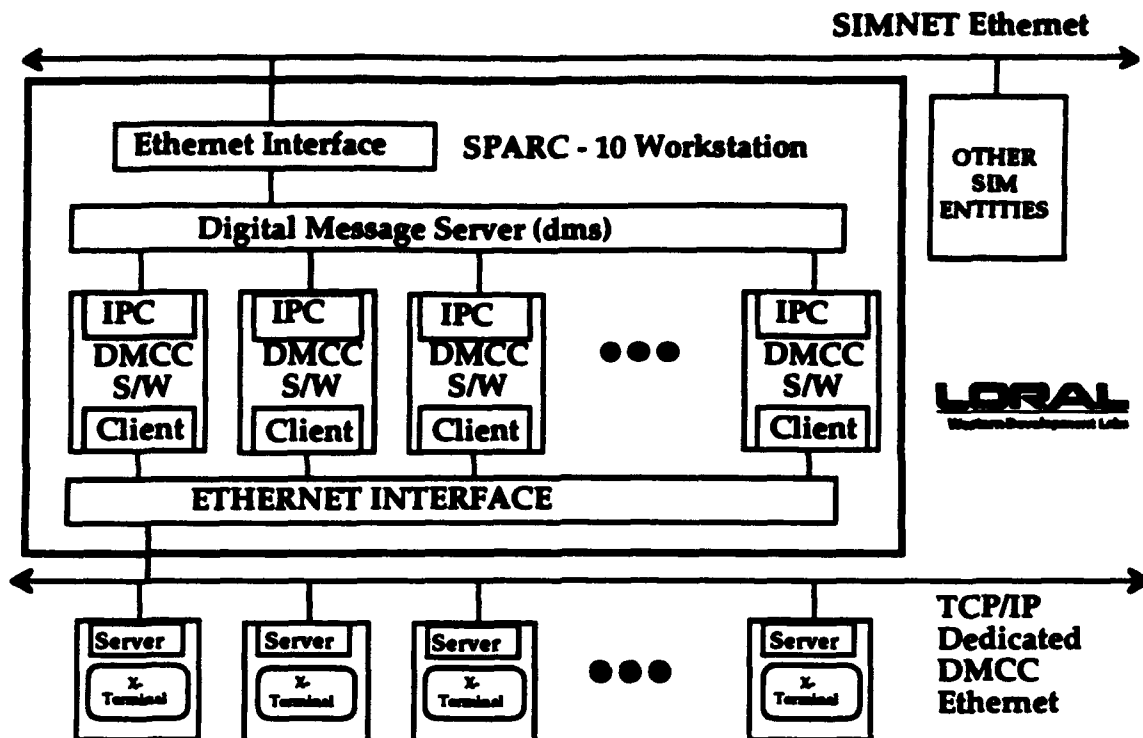


Figure 2 DMCC X-windows Client Server Environment

### 1.3 Document overview.

This document describes the system design of the Digital Message Communications Console CSCI. It outlines the structure and composition of the CSCI sub-functions (CSC's and CSU's) and provides a detailed description of each.

Section 2 describes the documents referenced in this specification.

Section 3 outlines the preliminary design overview of the CSCI, and a description of each CSC's purpose.

Section 4 describes the detailed design of each CSC.

Section 5 provides a detailed breakdown of all data elements defined for each CSC and for each associated CSU.

Section 6 provides a data file cross reference to aid in locating functions.

Sections 7 and 8 provide requirements and general design notes.

Section 9 contains structure charts for the overall DMCC CSCI and its component CSCs.

Section 10 details the datastructures.

Section 11 contains state transition diagrams for the various concurrent process CSCs in the DMCC CSCI.

Section 12 contains diagrams of the Protocol Data Units used by the Digital Message Communications Console to send and receive digital messages.

Section 13 covers the Functional Tests used to verify proper operation of the DMCC software.

## **2. Referenced and Relevant documents.**

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

### **2.1 Government documents.**

#### **SPECIFICATIONS:**

None.

#### **STANDARDS:**

DI-MISC-80711, Scientific and Technical Reports.

The SIMNET Network and Protocols, Report No. 7627, Arthur R. Pope;  
Prepared for DARPA by Bolt, Barenek and Newman, Inc. June 1991.

IST-CR-92-12

Military Standard: Draft DIS 2.0: Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation, Institute for Simulation and Training, 12424 Research Parkway, Suite 300, Orlando, Florida, September 4, 1992

### **2.2 Non-Government documents.**

Rotary Wing Aircraft AIRNET Aeromodel and Weapons Model Conversion Statement of Work, April 6, 1992.

Software Requirements & Interface Specification for the AIRNET MCC Comanche Support and Digital Message/Communications Upgrade,

System Specification for the RWA AIRNET Aeromodel and Weapons Conversion, LORAL Western Development Labs, 3200 Zanker Road, POBox 49041, San Jose, California, WDL/TR-92-003011, June 5, 1992.

The X Window System: Programming and Applications with Xt, OSF/Motif@ edition, Douglas A. Young, Hewlett-Packard Laboratories, Palo Alto, California, Prentice Hall, Englewood Cliffs, New Jersey, 1990

The Motif® Programming Manual for OSF/Motif version 1.1, volumes 1 - 6, (The Definitive Guides to the X-Window System) Dan Heller, O'Reilly & Associates, Inc., July 1992.

OSF/Motif® Programmer's Reference, Release 1.1, Open Software Foundation, Prentice Hall, Englewood Cliffs, New Jersey 1991

Builders Xcessory User's Guide, Version 2.0, Integrated Computer Solutions, Incorporated, 201 Broadway, Cambridge, Massachusetts, 1991.

Information technology--Local and metropolitan area networks--Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, International Standard ISO/IEC 8802-3 : 1992 ANSI/IEE Std 802.3, 1992 Edition, The Institute of Electronics and Electronics Engineers, Inc.

The Ethernet. Physical and Data Link Layer Specifications, Version 2.0. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation, 1982.

A Standard for the Transmission of IP Datagram over IEEE 802 Networks (Internet and Address Resolution Protocol on IEEE 802 Networks) Network Working Group, Request for comments(RFC): 1042 J. Postel and J. Reynolds, ISI, February 1988.



### **3. Preliminary design.**

#### **3.1 CSCI overview.**

The DMCC Computer Software Configuration Item includes all the software executeables to implement the Digital Message Communications Console. It includes CSC's for SIMNET communications; message routing; user interface, message storage, access, retrieval, preparation, and transmission; and message arrival notification.

Multiple X-client and X-client child processes may be instantiated to serve multiple X-terminal simulation entities.

The design of the Digital Message Communications Console software is detailed in this document.

### 3.1.1 CSCI architecture.

The structure charts shown in *Appendix A* outline the detailed organizational structure of the CSCI.

The following top level structure chart outlines the top level structure of the DMCC CSCI:

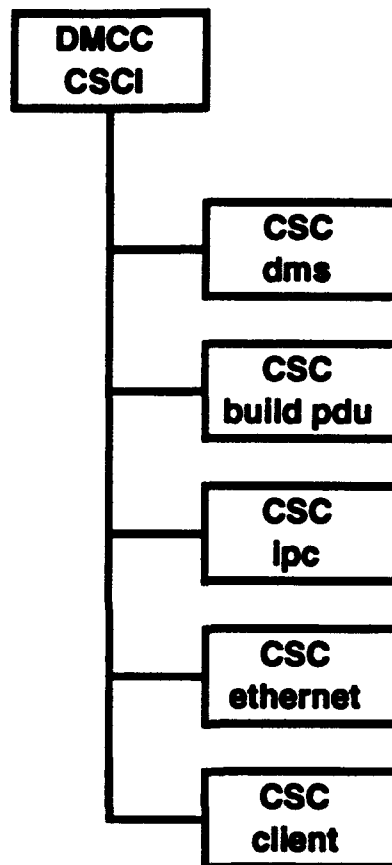


Figure 3 Top Level DMCC CSCs

The list of Computer Software Components (CSCs) and sub-level CSCs following this paragraph includes a summary of the purpose of each, the inter-CSC relationships and a CSC-to-CSC interface summary.

Table 1. CSC Summary table

CSC	Source File	Description/Interface Summary
dms	dms.c	<p><b>DESCRIPTION:</b> The dms function manages the distribution of messages to DMCC operators. It receives a login message from each operator, which contains the message CEOIs/Groups that should be sent to the operator.</p> <p><b>INPUTS:</b></p> <ol style="list-style-type: none"> <li>1. DMCC messages</li> <li>2. Login requests</li> <li>3. Logout requests</li> <li>4. CEOI/Group updates</li> </ol> <p><b>OUTPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Incoming shared memory segment</li> <li>2. Fan-out message queue</li> <li>3. Outgoing message queue</li> <li>4. Digital Message arrival notifications</li> </ol>
build pdu	bldpdu.c	<p><b>DESCRIPTION:</b> The build pdu function provides the X-window operator interface the means to construct Digital Messages and send them out to the SIMNET Ethernet process.</p> <p><b>INPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Digital Message data items</li> </ol> <p><b>OUTPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Digital Messages</li> <li>2. DTG</li> </ol>
ipc	dmslib.c	<p><b>DESCRIPTION:</b> The ipc function provides an interface for the X-window process, and the SIMNET Ethernet interface processes to send and receive DMCC messages. It initializes the X-window child process that receives new Digital Message arrival notifications from the dms process. The child process reads the new messages from shared memory and passes them to the X-window parent process via a pipe.</p> <p><b>INPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Digital Message from SIMNET interface process</li> <li>2. Digital Message data items from X-window operator interface process</li> <li>3. CEOI/Groups</li> </ol> <p><b>OUTPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Digital Message arrival notifications</li> <li>2. Digital Messages</li> <li>3. Login requests</li> <li>4. Logout requests</li> </ol>

MMI		<p><b>DESCRIPTION:</b> The MMI function allows the user to login, create, read and send messages</p> <p><b>INPUTS:</b></p> <ol style="list-style-type: none"> <li>1. DMCC messages</li> <li>2. Formatted messages for reading</li> </ol> <p><b>OUTPUTS:</b></p> <ol style="list-style-type: none"> <li>1. Login requests</li> <li>2. Logout requests</li> <li>3. CEOI/Group updates</li> <li>4. DMCC messages</li> <li>5. Digital Message data items</li> </ol>
Address list screen creation	addrList.c	<p><b>DESCRIPTION:</b> Creates Address List Screen</p> <p><b>INPUTS:</b> None</p> <p><b>OUTPUTS:</b> None</p>
Address list screen callbacks	addrListCB.c	<p><b>DESCRIPTION:</b> Address List Callbacks</p> <p><b>INPUTS:</b> None</p> <p><b>OUTPUTS:</b> None</p>
Console screen creation	console.c	<p><b>DESCRIPTION:</b> Creates Console Screen</p> <p><b>INPUTS:</b> None</p> <p><b>OUTPUTS:</b> None</p>
Console screen callbacks	consoleCB.c	<p><b>DESCRIPTION:</b> Console Screen callbacks</p> <p><b>INPUTS:</b> None</p> <p><b>OUTPUTS:</b> None</p>
Form trim drawing	formTrim.c	<p><b>DESCRIPTION:</b> Crop forms</p> <p><b>INPUTS:</b> None</p> <p><b>OUTPUTS:</b> None</p>

Free text report screen creation	freeTxt.c	DESCRIPTION: Creates free text msg prep screen INPUTS: None OUTPUTS: None
Free text report screen callbacks	freeTxtCB.c	DESCRIPTION: Free text report callbacks INPUTS: None OUTPUTS: None
Get DMCC messages	getMessageC B.c	DESCRIPTION: Retrieves messages from shared mem INPUTS: None OUTPUTS: None
Group list screen creation	groupList.c	DESCRIPTION: Creates group list screen INPUTS: None OUTPUTS: None
Group list screen callbacks	groupListCB.c	DESCRIPTION: Group list screen callbacks INPUTS: None OUTPUTS: None
List widget utilities	listUtilities.c	DESCRIPTION: List widget functionality INPUTS: None OUTPUTS: None
Location list screen creation	locList.c	DESCRIPTION: Creates location list screen INPUTS: None OUTPUTS: None

Location list screen callbacks	locListCB.c	DESCRIPTION: Location List Callbacks INPUTS: None OUTPUTS: None
Logon screen creation	logon.c	DESCRIPTION: Creates Logon screen INPUTS: None OUTPUTS: None
Logon screen callbacks	logonCB.c	DESCRIPTION: Logon screen callbacks INPUTS: None OUTPUTS: None
Main	main.c	DESCRIPTION: Main INPUTS: None OUTPUTS: None
Movcmd TMI screen callbacks	movcmdTmi_ call.c	DESCRIPTION: MOVCMDB screen callbacks INPUTS: None OUTPUTS: None
Movcmd TMI screen create	movcmdTmi_ create.c	DESCRIPTION: Creates MOVCMDB screen INPUTS: None OUTPUTS: None
Movcmd screen callbacks	movcmd_call. c	DESCRIPTION: MOVCMDB screen callbacks INPUTS: None OUTPUTS: None

Movcmd screen create	movcmd_create.c	DESCRIPTION: Creates MOVDMD screen INPUTS: None OUTPUTS:None
Message box screen create	msg1.c	DESCRIPTION: Creates Message Box Screen INPUTS: None OUTPUTS: None
Message box screen callbacks	msg1CB.c	DESCRIPTION: Message Box Screen Callbacks INPUTS: None OUTPUTS: None
Message read screen create	msgRead.c	DESCRIPTION: Creates READ screen INPUTS: None OUTPUTS: None
Message read screen callbacks	msgReadCB.c	DESCRIPTION: Read screen callbacks INPUTS: None OUTPUTS: None
MTO TMI screen callbacks	mtoTmi_call.c	DESCRIPTION: MTO screen callbacks INPUTS: None OUTPUTS: None
MTO TMI screen create	mtoTmi_create.c	DESCRIPTION: Creates MTO screen INPUTS: None OUTPUTS: None

MTO screen callbacks	mto_call.c	DESCRIPTION: MOT screen callbacks  INPUTS: None  OUTPUTS: None
MTO screen create	mto_create.c	DESCRIPTION: Creates MTO screen  INPUTS: None  OUTPUTS: None
Popup messages	popupMessage.c	DESCRIPTION: Executes popup messages  INPUTS: None  OUTPUTS: None
Reports screen create	report.c	DESCRIPTION: Creates Reports screen  INPUTS: None  OUTPUTS: None
Reports screen callback	reportCB.c	DESCRIPTION: Reports screen callbacks  INPUTS: None  OUTPUTS: None
Request TMI screen callbacks	reqtTmi_call.c	DESCRIPTION: Request screen callbacks  INPUTS: None  OUTPUTS: None
Request TMI screen create	reqtTmi_create.c	DESCRIPTION: Creates Request screen TMIs  INPUTS: None  OUTPUTS: None



Request screen callbacks	reqt_call.c	DESCRIPTION: Request screen callbacks INPUTS: None OUTPUTS: None
Request screen create	reqt_create.c	DESCRIPTION: Creates Request screen INPUTS: None OUTPUTS: None
Shot TMI screen callbacks	shotTmi_call.c	DESCRIPTION: Shot TMI screen callbacks INPUTS: None OUTPUTS: None
Shot TMI screen create	shotTmi_create.c	DESCRIPTION: Creates Shot screen TMIs INPUTS: None OUTPUTS: None
Shot screen callbacks	shot_call.c	DESCRIPTION: Shot screen callbacks INPUTS: None OUTPUTS: None
Shot screen create	shot_create.c	DESCRIPTION: Creates Shot screen INPUTS: None OUTPUTS: None
Splash TMI screen callbacks	splashTmi_call.c	DESCRIPTION: Splash TMI screen callbacks INPUTS: None OUTPUTS: None

Splash TMI screen create	splashTmi_create.c	DESCRIPTION: Creates Splash screen INPUTS: None OUTPUTS: None
Splash screen callbacks	splash_call.c	DESCRIPTION: Splash screen callbacks INPUTS: None OUTPUTS: None
Splash screen create	splash_create.c	DESCRIPTION: Creates splash screen INPUTS: None OUTPUTS: None
Spot TMI screen callbacks	spotTmi_call.c	DESCRIPTION: Spot TMI screen callbacks INPUTS: None OUTPUTS: None
Spot TMI screen create	spotTmi_create.c	DESCRIPTION: Creates spot screen TMIs INPUTS: None OUTPUTS: None
Spot screen callbacks	spot_call.c	DESCRIPTION: Spot screen callbacks INPUTS: None OUTPUTS: None
Spot screens create	spot_create.c	DESCRIPTION: Creates Spot screen INPUTS: None OUTPUTS: None

Main screen create	sysMain.c	DESCRIPTION: Creates SYS MAIN screen  INPUTS: None  OUTPUTS: None
Main screen callbacks	sysMainCB.c	DESCRIPTION: Main screen callbacks  INPUTS: None  OUTPUTS: None
MMI utilities	util.c	DESCRIPTION: Various Man Machine Interface Utilities  INPUTS: Various (See section 4)  OUTPUTS: Various (See section 4)

### 3.1.2 System states and modes.

The DMCC software consists of five concurrent process types, to receive, route and transmit ethernet Protocol Data Units, and the X-Client and X-Client Child processes. All of these process types, with the exception of the X-Client, are modeless, that is, they wait for a single type of event and then handle that event in a single-threaded, one shot manner.

The X-Client Process, on the other hand, has many states or modes, and actions which this process takes in response to events depend on many factors. The basic modes or states which the DMCC Client Process can exhibit are closely bound to the concept of an "active" X-Window.

The X-Client Process State Event Matrix is shown in Appendix C. This diagram shows actions and transitions from various window states bound to event flags representing operator intervention, such as clicking on a button region of a window. Each cell shows what actions are taken as a result of an event followed by a semicolon (;) character, which is followed by an indication of the next window to transition to (make active), if any. The top row is a list of the window events and the left column is a list of window states.

### 3.1.3 Memory and processing time allocation.

Because the Digital Message Communication System is a near-real-time messaging system, there are no hard requirements for processing time, other than to respond fast enough such that seamless "realistic" simulation is maintained.. The Sparc-10 workstation platform provides processing speed (56,000,000 instructions per second) an order of magnitude in excess of that required for timely user interface activity and message handling.

The following table depicts the rough-order-of-magnitude, worst case memory requirements for the DMCC software suite:

Table 2      Memory Requirements

Process	Rough Order Of Magnitude, Worst Case Memory Requirement
System	16 MB
8 X window managers	12 MB
dms process	1 MB
8 Client Processes	12 MB
8 Client Children	1 MB
Ethernet Receive	1 MB
Ethernet Transmit	1 MB
Total	44 MB

### **3.2 CSCI design description**

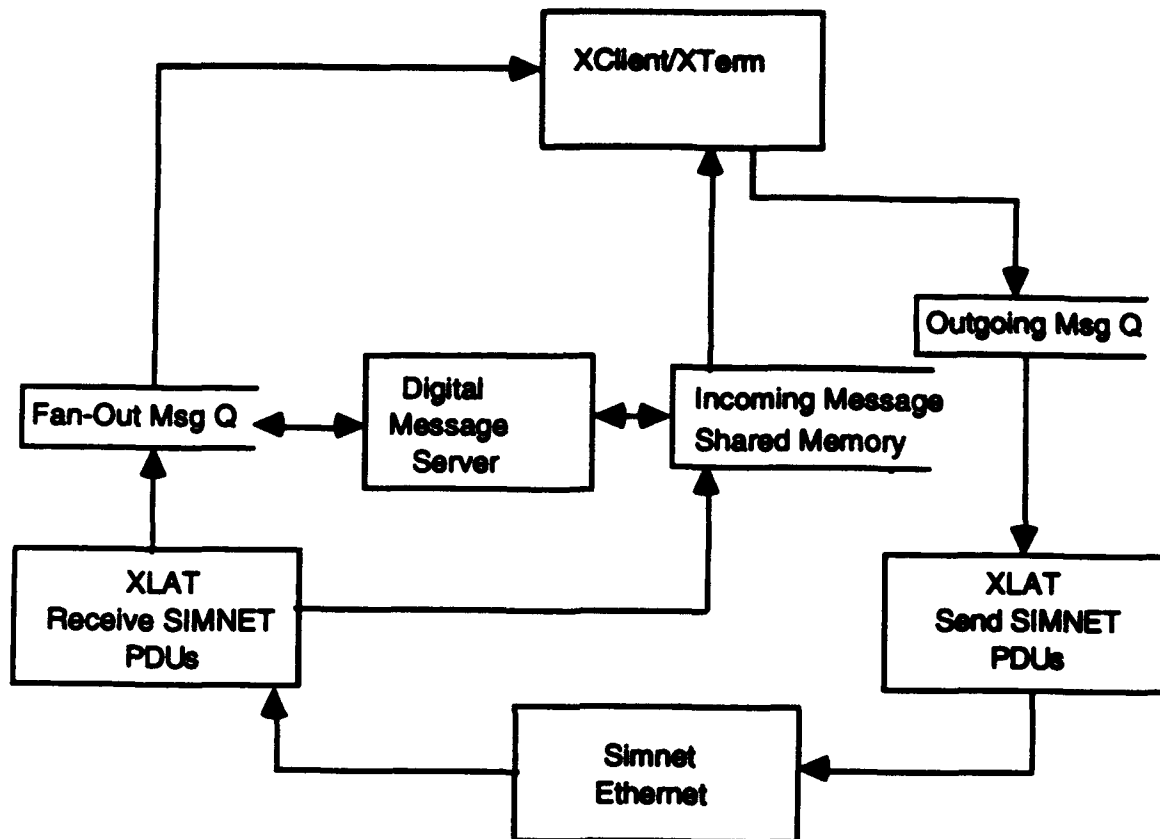
The DMCC CSCI consists of five concurrent processes to receive ethernet messages (Receive), rout them to client DMCC entities (Digital Message Server) , to transmit ethernet messages (Transmit), to handle user interface and manage user message queues (X-Client) and to inform the X-Client of message arrival (X-Client\_Child). In a particular installation, only one Transmit, Receive and Digital Message Server process is active at any one time. The X-Client and X-Client Child processes are instantiated for each of the active Digital Message Communications Consoles (X-Terminals) active in the system. Thus, for an eight client system, a total of nineteen concurrent processes may be active.

#### **3.2.1 CSC dms**

DESCRIPTION: The dms function manages the distribution of messages to DMCC operators. It receives a login message from each operator, which contains the message CEOs/Groups that should be sent to the operator.

The following diagram shows the relationship of the digital message server to the other concurrent processes in the DMCC software.

**Block Diagram of DMCC  
(10/8/82 v2)**  
(Each box corresponds to a process)



**Figure 4 DMCC Concurrent Processes**

Specific duties performed by the digital message server are:

1. Remove any existing copies of shared memory segment and message queues.
2. Create Incoming shared memory segment, Fan-Out message queue, and Outgoing message queue.
3. Process login and logout requests from X-Client CSC.
4. Store CEOIs/Groups received in login message or newdests message from X-Client CSC.
5. Process new DMCC PDU arrival notifications from ethernet CSC.
6. Determine which operators should be notified of the arrival of a new DMCC PDU based on a match between the Target CEOI in the message, and the CEOIs/Groups passed in the login message or newdests message, and the exercise id contained in the new DMCC PDU and the exercise that was passed from the operator in the login message.

7. Send kill signal via operating system interface to the X-client child when an operator logs-out and then remove all notifications sent to operator in Fan-Out queue, and decrement 'in use' (this is data item "nbr\_readers" in the shared memory segment) flag for each message in shared memory segment.

The following State Transition Diagram details the activity sequencing for the digital message server:

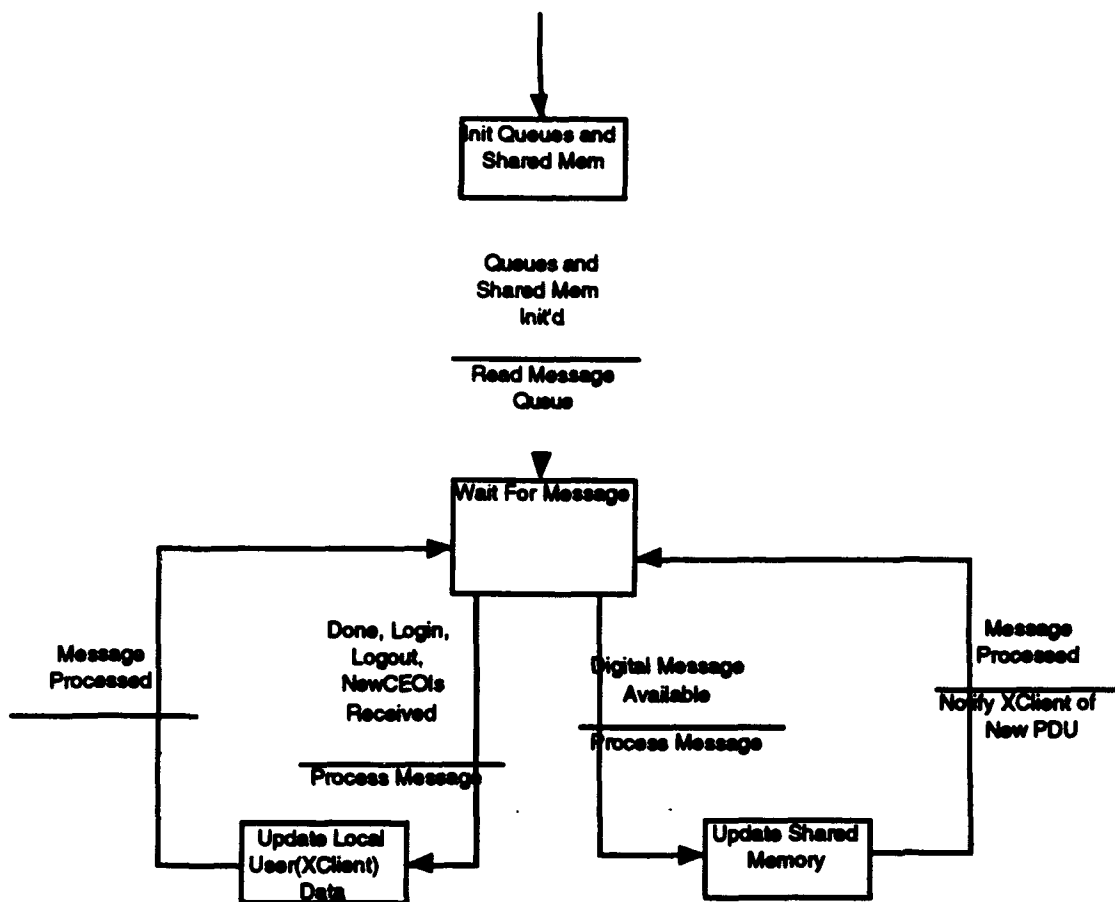


Figure 5 CSC dms State Transition Diagram

#### CSC INTERFACES:

1. "shmaddr" data item is initialized with address of shared memory segment. Offsets from this address contain messages and are passed to CSC client when a new message arrives.
2. Data item "msgin" with flag msg\_type set to MSG\_DMAVAIL is received from CSC ethernet and contains notification of the arrival of a new message, and its location in the shared memory segment.
3. Data item "msgin" with flag msg\_type set to MSG\_DONE is received from CSC client and indicates when a message has been read.

4. Data item "msgin" with flag msg\_type set to MSG\_LOGOUT is received from CSC client and indicates a logout.
5. Data item "msgin" with flag msg\_type set to MSG\_LOGIN is received from CSC client and indicates an operator is logging-in.
6. Data item "msgin" with flag msg\_type set to MSG\_NEWDESTS is received from CSC client and indicates that the operators list of CEOs/Groups has changed.
7. Data item "msgout" is sent to CSC client when a new message is available.

### 3.2.2 CSC build pdu

**DESCRIPTION:** The build pdu function provides the X-window operator interface process the means to construct Digital Messages and send them out to the SIMNET Ethernet interface process.

Specific duties performed are:

1. Create the appropriate Digital Message.
2. Send the message to CSC ethernet.
3. Return the current Zulu DTG.
4. Forward a received message to another operator.

#### CSC INTERFACES:

1. Data item "pdu" is sent to the CSC ethernet via the Outgoing queue
2. Each PDU builder CSU has it's own interface which includes those unique data items needed to build the PDU. These are: TBD
3. Data item "dtg" is returned to caller CSC client containing Zulu date-time group.
4. Data item "dmcc\_timezone\_offset\_g" contains the localtime to zulu time difference in minutes, and is received from CSC client via the file "/tmp/dmcc\_timezone\_offset".

### 3.2.3 CSC ipc

**DESCRIPTION:** The ipc function provides an interface for the X-window process, and the SIMNET Ethernet interface processes to send and receive DMCC messages. It initializes the X-window client child process that receives new Digital Message arrival notifications from the dms process. The child process reads the new messages from shared memory and passes them to the X-window parent process via a pipe.

Specific duties performed are:

1. Create a 'child' process of CSC client that is used to communicate with CSC dms.
2. Attach to shared memory segment and Fan-Out queue for CSC client 'child'.
3. Attach to Outgoing queue for CSC dmcc 'parent'.



4. Attach to Fan-Out queue for CSC dmcc\_sim\_tx.
5. Send login and logout message to CSC dms.
6. Put a message in the Outgoing queue.
7. Receive a message from the Fan-Out queue containing notification of the arrival of a new DMCC PDU.
8. Find the address of a free record in the shared memory segment.
9. Copy a DMCC PDU into the shared memory segment.
10. Send updated lists of operator CEOI's/Group's from CSC dmcc to CSC dms.
11. Read the pipe between the X-client process and the child process.
12. Return the exercise id of the operator to a caller.
13. Return the operators name to the caller.

#### CSC INTERFACES:

1. Data item "mode" received from CSC's client and ethernet.
2. Data item "buf" received from CSC client .
3. Data item "buflen" received from CSC client .
4. Data item "buf" passed to CSC client .
5. Data item "getfreerecord" passed to CSC ethernet.
6. Data item "new\_dm\_address" is received from CSC ethernet.
7. Data item "buf" is received from CSC ethernet
8. Data item "buflen" received from CSC ethernet.
9. Data item "dests" received from CSC ethernet.
10. Data item "exid" received from CSC ethernet.
11. Data item "dests" received from CSC client .
12. Data item "exid" received from CSC client .
13. Data item "ceoi" received from CSC client .
14. Data item "p" received from CSC client .
15. Data item "pdu" received from CSC client .
16. Data item "dms\_getexid" passed to CSC build pdu.
17. Data item "dms\_getopername" passed to CSC build pdu.
18. Data item "dmslogin" passed to CSC client .

The following diagram depicts the state transitions of the Client Child process:

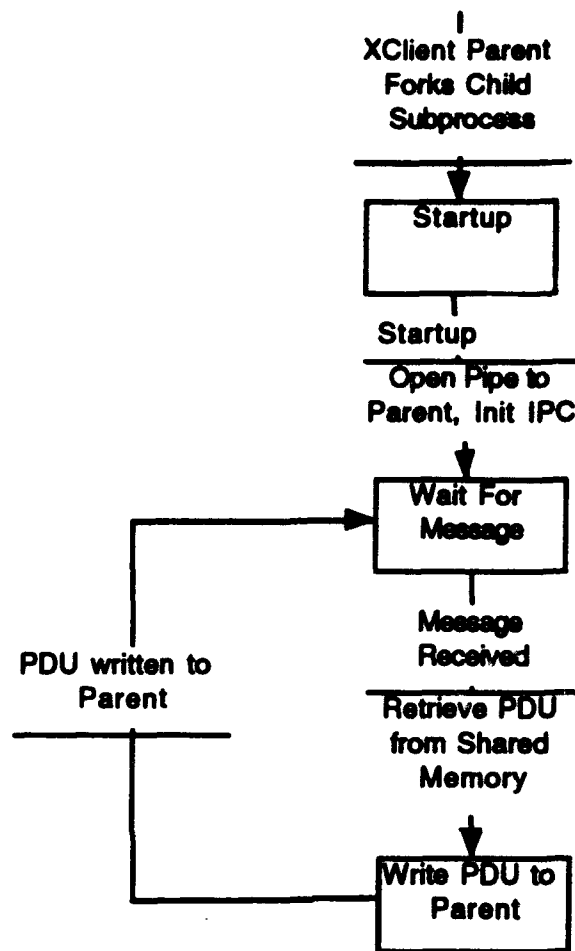


Figure 6 Client Child State Transition Diagram

### 3.2.4 CSC ethernet

**DESCRIPTION:** The ethernet CSC contains the functionality for receiving DMCC PDUs from and sending them to the SIMNET ethernet network. It consists of two concurrent processes, one for transmission and one for reception of messages.

Specific duties performed are:

1. Set up receive and transmit queues for the incoming and outgoing messages.
2. Perform encode/decode actions on the messages and completeness checks on the contents of those messages.

**CSC INTERFACES:**

1. Name of computer interface is passed to dmcc\_sim\_rx and dmcc\_sim\_tx via operator input:

dmcc\_sim\_rx *Ethernet\_interface\_name*

dmcc\_sim\_tx *Ethernet\_interface\_name*

**3.2.4.1 Sub-level CSC dmcc\_sim\_rx process**

**DESCRIPTION:** The dmcc\_sim\_rx function initializes the message queue, invokes functions to initialize ethernet interface to receive messages in promiscuous mode on Ethernet, to read in message, to timestamp the input message, to validate the SNAP and association layer headers of the input message, to validate exercise identification number, to check for valid association pdu format, to split up the input message packet into individual association PDUs if applicable, and to copy parsed/checked message onto CSC dms shared memory.

Specific duties performed are:

1. get ethernet interface name from command line argument and copy it onto the global variable SIM\_interface\_name.
2. initialize message queue with dmsinit().
3. invoke SIM\_rx\_netif().

**CSC INTERFACES:**

1. on line command argument "le0" or "le1" is the interface name and is copied onto the global variable SIM\_interface\_name.
2. Data Item "DMS\_ENET\_TRANSMIT" is passed to CSU dmsinit.

**3.2.4.2 Sub-level CSC dmcc\_sim\_tx**

**DESCRIPTION:** The dmcc\_sim\_tx function controls the flow of messages from the DMCC onto the Ethernet. It parses the interface name from on-line command argument and invokes functions to read pdu from initialized message queue, to add association layer and padding bytes to the input message if applicable, to initialize ethernet interface to send messages onto the network as broadcast messages, to set the IEEE 802.3 MAC sublayer frame control header with the destination broadcast address, to fill the IEEE 802.2 LLC sublayer frame control header, to add padding bytes if necessary, and to output message onto Ethernet.

Specific duties performed are:

1. Initialize message queues for each of the input Ethernet interface names.
2. Zero the message buffer and read a new message into the buffer.
3. Send the message to the pdu encode function.

**CSC INTERFACES:**

1. on line command argument "le0" or "le1" is the interface name and is copied onto the global variable SIM\_interface\_name.
2. Data item "DMS\_ENET\_TRANSMIT" is passed to CSU dmsinit.
3. Data item "input buffer- buff" is passed to CSU bzero.
4. Data item "input buffer-buff" is passed to CSU dms\_recvdm.
5. Data items "NEW\_ALPDU-pdu sequence", "input buffer-buff" and "number bytes read-nbyte\_read" passes to CSU ALPDU\_SIM\_encode.

**3.2.5 CSC client**

**DESCRIPTION:** The client function provides the interface with the user, allowing the user to create and read Digital Messages.

Specific duties performed are:

1. Allow the user to login and send the login information to the dms CSC.
2. Allow the user to enter CEOI/Group information and send it to the dms CSC.
3. Receive PDU from the dms CSC. The client will then send the PDU to the PDU decode CSC for translation to a user readable form. This message is then placed in a global structure and an entry is added to the MSG1 screen and the New Mail icon is updated.
4. Allows the user to read the message.
5. Allows the user to reply to the message.
6. Allows the user to forward the message.
7. Allows the user to forward the message and include an additional message.
8. Allows the user to delete the message from the message queue.
9. Allows the user to compose a Digital Message and send it either as an urgent or a routine message.

**CSC INTERFACES:**

1. Data item "readBuf" is received from dms CSC via a pipe with the label pipeFD. This data item is then placed in the global data item "messages" as the "pdu" field for use later.
2. The "pdu" field of the data item "messages" is passed to pdu\_decode.
3. The data item "msgBuf" is returned from the call to pdu\_decode.
4. Data item "dmcc\_timezone\_offset\_g" contains the localtime to zulu time difference in minutes and is received from a read of the file "/tmp/dmcc\_timezone\_offset\_g".
5. The data item "reusePDU" is sent to the bldpdu CSC through a call to fwdpdu.
6. The data item "SenderCEOI" containing the name of the sender of the Digital Message is passed to the bldpdu CSC through a call to bldAck. This call sends an acknowledgment pdu to the sender.

7. Data items (TBD) are sent to the bldpdu CSC via one of the bldpdu functions, depending on what type of Message.

### 3.2.6 CSC pdu\_decode

**DESCRIPTION:** The pdu decode function decodes the input pdu based on its specific message type and returns the decoded message to caller. It outputs error message for unrecognizable message type.

Specific duties performed are:

1. Decode the input message header.
2. Decode the input message based on its message type.
3. Concatenate the decoded message header and body.
4. Return the decoded message to caller.

#### CSC INTERFACES:

1. Data item "pdu" pass-in to CSC pdu\_decode.
2. Data item "pdu" is passed to the CSU pdu\_spec\_decode for decoding the header information.
3. Data item "pdu" is casted with its specific message type before passing to the appropriate message decode function to decode the message body.

Possible message decode functions:

```
decode_ack_pdu( (Ack_type *) pdu)
decode_spot_pdu( (Spot_type *) pdu)
decode_bda_pdu( (BDA_type *) pdu);
decode_free_text_pdu( (FreeText_type *) pdu)
decode_gndroute_pdu( (Recon_type *) pdu)
decode_airroute_pdu( (Recon_type *) pdu)
decode_bridge_pdu( (Recon_type *) pdu)
decode_lzpz_pdu( (Recon_type *) pdu)
decode_bpop_pdu( (Recon_type *) pdu)
decode_crossing_pdu( (Recon_type *) pdu)
decode_repeat_pdu( (Artillery_type *) pdu)
decode_cancel_pdu( (Artillery_type *) pdu)
decode_check_pdu( (Artillery_type *) pdu)
decode_cno_pdu( (Artillery_type *) pdu)
decode_shift_pdu( (Artillery_type *) pdu)
decode_nwmsn_pdu( (Artillery_type *) pdu)
decode_mto_pdu( (Artillery_type *) pdu)
decode_shot_pdu( (Artillery_type *) pdu)
decode_splash_pdu( (Artillery_type *) pdu)
decode_eom_pdu( (Artillery_type *) pdu)
decode_status_pdu( (Status_type *) pdu)
decode_request_pdu( (Request_type *) pdu)
decode_nbc1_pdu( (NBC_type *) pdu)
decode_nbc4_pdu( (NBC_type *) pdu)
```

```
decode_nbc5_pdu( (NBC_type *) pdu)
decode_miji_pdu( (MIJI_type *) pdu)
decode_pirep_pdu( (PIREP_type *) pdu)
decode_dnav_pdu( (DNAV_type *) pdu)
decode_movcmd_pdu( (Move_type *) pdu)
```

4. Data item "dmc\_buffer" which contains the decoded message header and body is being returned to caller.

#### 4. Detailed design.

The following section identifies and describes the relationship of each CSU to its calling CSC. The table below provides a reference of CSU's and their calling CSC's.

Table 3. CSC/CSU Cross Reference Table.

CSC	CSU
dms	main
	init_queue
	init_shared_mem
	wait_for_msg
	cmpfixedstr
	trimup
	cleanup_xchild
build pdu	setdtg
	bldAck
	bldFreeText
	bldSpot
	bldRequest
	bldMove
	bldMTO
	bldShot
	bldSplash
	fwdpdu
ipc	dmsinit
	dms_senddm
	dms_recvdm
	dms_getfreerecord
	dms_seekrecord
	dms_notify
	dms_bcopy
	dmslogin
	dmsnewdests
	dmslogout
	dms_getdm
	dms_getexid
	dms_getopername

#### 4.1 CSC dms.

The following paragraphs under this section describe the relationship of the dms CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the dms CSC.

CSC dms satisfies System Segment Specification requirements 3.2.1.2.1 and 3.2.1.2.2.

The following structure chart outlines the top level structure of the dms CSC.

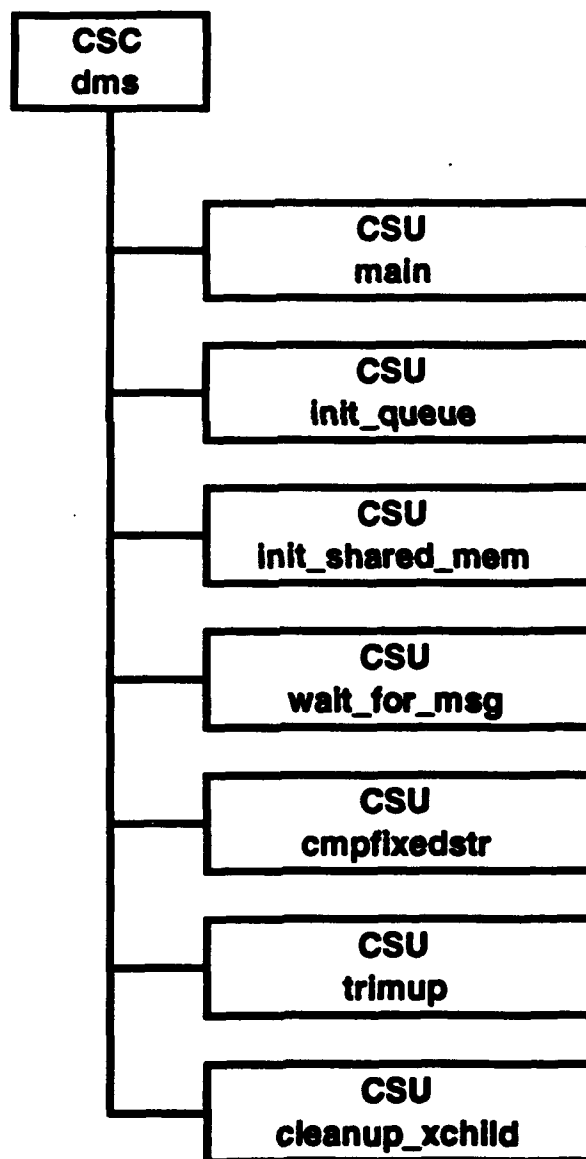


Figure 6a CSC dms Structure



#### 4.1.1 CSU main.

The main CSU is a driver for the dms function, and calls routines to initialize message queues and shared memory, and then calls a routine that infinitely waits for message notifications to arrive. The following paragraphs provide design information for this CSU.

##### 4.1.1.1 CSU main design specification/constraints.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

a. Input/output data elements.

INPUTS:

None

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the main CSU:

"fanout\_qid" stores the id of the Fanout message queue.

"outgoing\_qid" stores the id of the Outgoing queue.

"shmaddr" stores the address of the shared memory segment.

h. Logic flow. Single pass.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.1.2 CSU init\_queue.

The init\_queue CSU creates the message queues used in the DMCC. It will first delete any queues left from a previous run of the software. The following paragraphs provide design information for this CSU.

##### 4.1.2.1 CSU init\_queue requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.2.2 CSU init\_queue design.

The information identified below represents the detailed design of the init\_queue CSU.

- a. Input/output data elements.

###### INPUTS:

"key" is a data item containing the operating system if of the queue to be created.

OUTPUTS: None.

- b. Local data elements.

"qid" is a data item used to store the id of the created message queue.

- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. logs an error if the queues cannot be attached, and returns 1
- f. Data conversion. None

- g. Use of other elements. Other elements that are used by the init\_queue CSU: None
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.1.3. CSU init\_shared\_mem.

The init\_shared\_mem CSU creates the shared memory segment used in the DMCC. It will first delete the segment if one is left from a previous run of the software. The following paragraphs provide design information for this CSU.

##### 4.1.3.1 CSU init\_shared\_mem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.3.2 CSU init\_shared\_mem design.

The information identified below represents the detailed design of the init\_shared\_mem CSU.

- a. Input/output data elements.

INPUTS:

"key" is the operating system id for the shared memory segment.

"size" is the length in bytes of the segment.

OUTPUTS:

"init\_shared\_mem" is the address of the shared memory segment.

- b. Local data elements.

"mid" stores the id of the shared memory segment.

- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. logs an error if the shared memory segment cannot be attached, and returns 1.
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the bzero CSU: None
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.1.4. CSU wait\_for\_msg.

The wait\_for\_msg CSU waits for new messages in the Fan-Out queue. These messages are from the CSC dmcc\_sim\_rx with notifications of a new DMCC PDU arrival, or from CSC dmcc with login, logout, message read, or new CEOs/Groups. The following paragraphs provide design information for this CSU.

##### 4.1.4.1 CSU wait\_for\_msg requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.4.2 CSU wait\_for\_msg design.

The information identified below represents the detailed design of the wait\_for\_msg CSU.

- a. Input/output data elements.

INPUTS: None

OUTPUTS: None

- b. Local data elements.  
"maxshmem" contains the highest address in the shared memory segment
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. None
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the wait\_for\_msg CSU:  
"fanout\_qid"  
"msgin"  
"dm\_rqsts"  
"shmaddr"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.1.5 CSU cmpfixedstr.

The cmpfixedstr CSU compares 2 vectors of strings and returns true if any string in one matches one in the other. The following paragraphs provide design information for this CSU.

##### 4.1.5.1 CSU cmpfixedstr requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.1.5.2 CSU cmpfixedstr design.**

The information identified below represents the detailed design of the cmpfixedstr CSU.

a. Input/output data elements.

INPUTS:

"ain" is a pointer to a vector of strings

"bin" is a pointer to a vector of strings

"size" is the maximum length of each string

"nbtocompare" is a count of the number of strings to compare

OUTPUTS:

"cmpfixedstr" is 1 if there is a match, or 0 if there is no match

b. Local data elements.

"i","j" are local indices

"alen","blen" are the lengths of each string being compared

"a","b" are pointers that hold the address of the current strings being compared

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if no match, otherwise 1

f. Data conversion. None

g. Use of other elements. Other elements that are used by the cmpfixedstr CSU: None

h. Logic flow. 2 nested for loops running over each vector of strings.

i. Data structures. None

j. Local data files or database. None

k. Limitations. CEOI strings must be no larger than "size" or terminate with a NULL

#### 4.1.6 CSU trimup.

The trimup CSU trims trailing, leading and embedded spaces, and upcases all letters, in a fixed size string. The following paragraphs provide design information for this CSU.

##### 4.1.6.1 CSU trimup requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.6.2 CSU trimup design.

The information identified below represents the detailed design of the trimup CSU.

a. Input/output data elements.

INPUTS:

"a" is a pointer to a string

"size" is the size of the string

OUTPUTS:

"a" is a pointer to a string--the trimmed and upcased string overwrites the input string

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the trimup CSU: None

h. Logic flow. For loop over each character in string.

i. Data structures. None

- j. Local data files or database. None
- k. Limitations. CEOI strings must be no larger than "size" or terminate with a NULL

#### 4.1.7 CSU cleanup\_xchild.

The cleanup\_xchild CSU cleans up after an XClient-child logs-out or is noticed to have disappeared. It sends SIGHUP and SIGKILL to the child process, and then tries to retrieve any messages left in the Fan-Out queue for that process. If any messages are retrieved then the "shmem\_addr" pointer is used to locate messages destined for the process, and decrements the "nbr\_readers" counter. The following paragraphs provide design information for this CSU.

##### 4.1.7.1 CSU cleanup\_xchild requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.1.7.2 CSU cleanup\_xchild design.

The information identified below represents the detailed design of the cleanup\_xchild CSU.

- a. Input/output data elements.

INPUTS:

"id" is the process id of the X-Client child process

OUTPUTS: None

- b. Local data elements.

None

- c. Interrupts and signals. None

- d. Algorithms. None

- e. Error handling. None

- f. Data conversion. None



- g. Use of other elements. Other elements that are used by the cleanup\_xchild CSU:  
"fanout\_qid"
- h. Logic flow. While loop over all messages retrieved from queue.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.2. CSC build pdu.

The following paragraphs under this section describe the relationship of the build pdu CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the build pdu CSC.

The following diagram details the top level structure of the build\_pdu CSC:

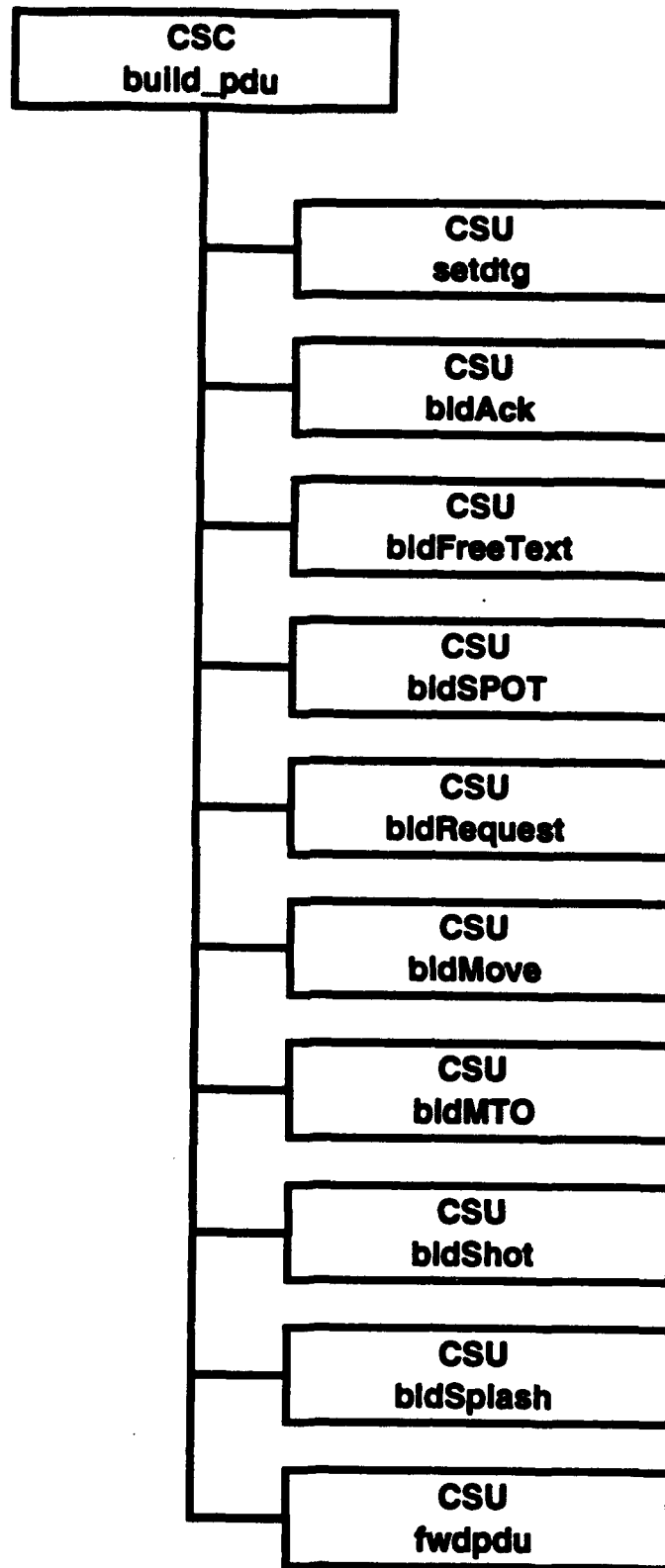


Figure 7 build\_pdu Structure

#### 4.2.1 CSU setdtg.

The setdtg CSU returns the current Zulu date-time group to the caller. The following paragraphs provide design information for this CSU.

##### 4.2.1.1 CSU setdtg requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.2.1.2 CSU setdtg design.

The information identified below represents the detailed design of the setdtg CSU.

a. Input/output data elements.

INPUTS:

"dtg" is a pointer to a data store for the dtg

OUTPUTS:

"dtg" is a pointer to a data store for the dtg

b. Local data elements.

"tp" contains current time

"tm" contains localtime

"buf" contains formatted dtg string

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the setdtg CSU: None

h. Logic flow. Single pass.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.2.2 CSU bldAck.

The bldAck CSU builds and sends an Ack Digital Message. The following paragraphs provide design information for this CSU.

##### 4.2.2.1 CSU bldAck requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.2.2.2 CSU bldAck design.

The information identified below represents the detailed design of the bldAck CSU.

- a. Input/output data elements.

INPUTS:

"targetCEOI" is a pointer to destination CEOI

OUTPUTS: None

- b. Local data elements.

None

- c. Interrupts and signals. None

- d. Algorithms. None

- e. Error handling. None

- f. Data conversion. None

- g. Use of other elements. Other elements that are used by the bldAck CSU:  
"pdu"

- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.2.3 CSU bldFreeText.**

The bldFreeText CSU builds and sends a FreeText Digital Message. The following paragraphs provide design information for this CSU.

##### **4.2.3.1 CSU bldFreeText requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.2.3.2 CSU bldFreeText design.**

The information identified below represents the detailed design of the bldFreeText CSU.

- a. Input/output data elements.

**INPUTS:**

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"FreeText" is a pointer to the free text string

**OUTPUTS:** None

- b. Local data elements.

None

- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. None

- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the bldFreeText CSU:  
"pdu"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.2.4 CSU bldSpot.

The bldSpot CSU builds and sends the SPOT Digital Message. The following paragraphs provide design information for this CSU.

##### 4.2.4.1 CSU bldSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.2.4.2 CSU bldSpot design.

The information identified below represents the detailed design of the bldSpot CSU.

##### a. Input/output data elements.

###### INPUTS:

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"TimeSighted" is a pointer to the time sighted dtg

"ObeserverLocation" is a pointer to the observers UTM

"TargetQuantity" is a count of the targets

"ObserverIntentions" is a byte indicating the observers intentions

"TargetType" is a byte indicating the type of the target

"TargetActivity" is a byte indicating the activity of the target  
"TargetDirection" is a byte indicating the direction of the target  
"TargetSpeed" is a pointer to the speed of the target  
"TargetLocation" is a pointer to the target's UTM  
"TargetUnit" is a byte indicating the unit of the target

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the  
bldSpot CSU:  
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.2.5 CSU bldRequest.

The bldRequest CSU builds and sends the Request Digital Message. The following paragraphs provide design information for this CSU.

##### 4.2.5.1 CSU bldRequest requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.2.5.2 CSU bldRequest design.

The information identified below represents the detailed design of the bldRequest CSU.

a. Input/output data elements.

INPUTS:

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"ReportType" is a byte indicating the type of report

"ReconType" is a byte indicating the type of RECON

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldRequest CSU:  
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.2.6 CSU bldMove.

The bldMove CSU builds and sends Move Digital Messages. The following paragraphs provide design information for this CSU.



**4.2.6.1 CSU bldMove requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.2.6.2 CSU bldMove design.**

The information identified below represents the detailed design of the bldMove CSU.

**a. Input/output data elements.****INPUTS:**

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"TargetID" is a pointer to the id of the target

"Task" is a byte indicating the task

"Who" is a byte indicating who

"When" is a byte indicating when

"DTG" is a pointer to a dtg

**OUTPUTS:** None

**b. Local data elements.**

None

**c. Interrupts and signals. None****d. Algorithms. None****e. Error handling. None****f. Data conversion. None****g. Use of other elements. Other elements that are used by the bldMove CSU:**  
"pdu"**h. Logic flow. Single pass.**

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.2.7 CSU bldMTO.

The bldMTO CSU builds and sends MTO Digital Messages. The following paragraphs provide design information for this CSU.

##### 4.2.7.1 CSU bldMTO requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.2.7.2 CSU bldMTO design.

The information identified below represents the detailed design of the bldMTO CSU.

- a. Input/output data elements.

INPUTS:

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"MissionStatus" is a byte indicating the status of the mission

"RequestAdjustment" is a byte indicating an adjustment request

"EnterTarget" is a byte indicating the target to enter

"EndMission" is a byte indicating the mission end

OUTPUTS: None

- b. Local data elements.

None

- c. Interrupts and signals. None

- d. Algorithms. None

- e. Error handling. None
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the bldMTO CSU:  
"pdu"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.2.8 CSU bldShot.**

The bldShot CSU builds and sends Shot Digital Messages. The following paragraphs provide design information for this CSU.

##### **4.2.8.1 CSU bldShot requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.2.8.2 CSU bldShot design.**

The information identified below represents the detailed design of the bldShot CSU.

##### **a. Input/output data elements.**

###### **INPUTS:**

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"MissionStatus" is a byte indicating the status of the mission

"RoundsFired" is a byte indicating the rounds fired

"ImpactTime" is a byte indicating the time of impact

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the bldShot CSU:  
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.2.9. CSU bldSplash.

The following paragraphs provide design information for this CSU.

##### 4.2.9.1 CSU bldSplash requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.2.9.2 CSU bldSplash design.

The information identified below represents the detailed design of the bldSplash CSU.

a. Input/output data elements.

INPUTS:

"targetCEOI" is a pointer to the destination CEOI strings

"annotation" is a pointer to the annotation string

"prio" is the PDU priority

"MissionStatus" is a byte indicating the status of the mission

"RoundsFired" is a byte indicating the rounds fired

"ImpactTime" is a byte indicating the time of impact

OUTPUTS: None

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the  
bldSplash CSU:  
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.2.10 CSU fwdpdu.

The fwdpdu CSU builds and sends forwarded Digital Messages. The following paragraphs provide design information for this CSU.

**4.2.10.1 CSU fwdpdu requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.2.10.2 CSU fwdpdu design.**

The information identified below represents the detailed design of the fwdpdu CSU.

a. Input/output data elements.

INPUTS:

"fwdpdu" is a pointer to the destination CEOI string

"tpdu" is a pointer to a pdu

OUTPUTS: None

b. Local data elements.

"pdu\_size" contains the size of the pdu

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the fwdpdu CSU:  
"pdu"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

### 4.3 CSC ipc

The following diagram details the top level structure of the ipc CSU:

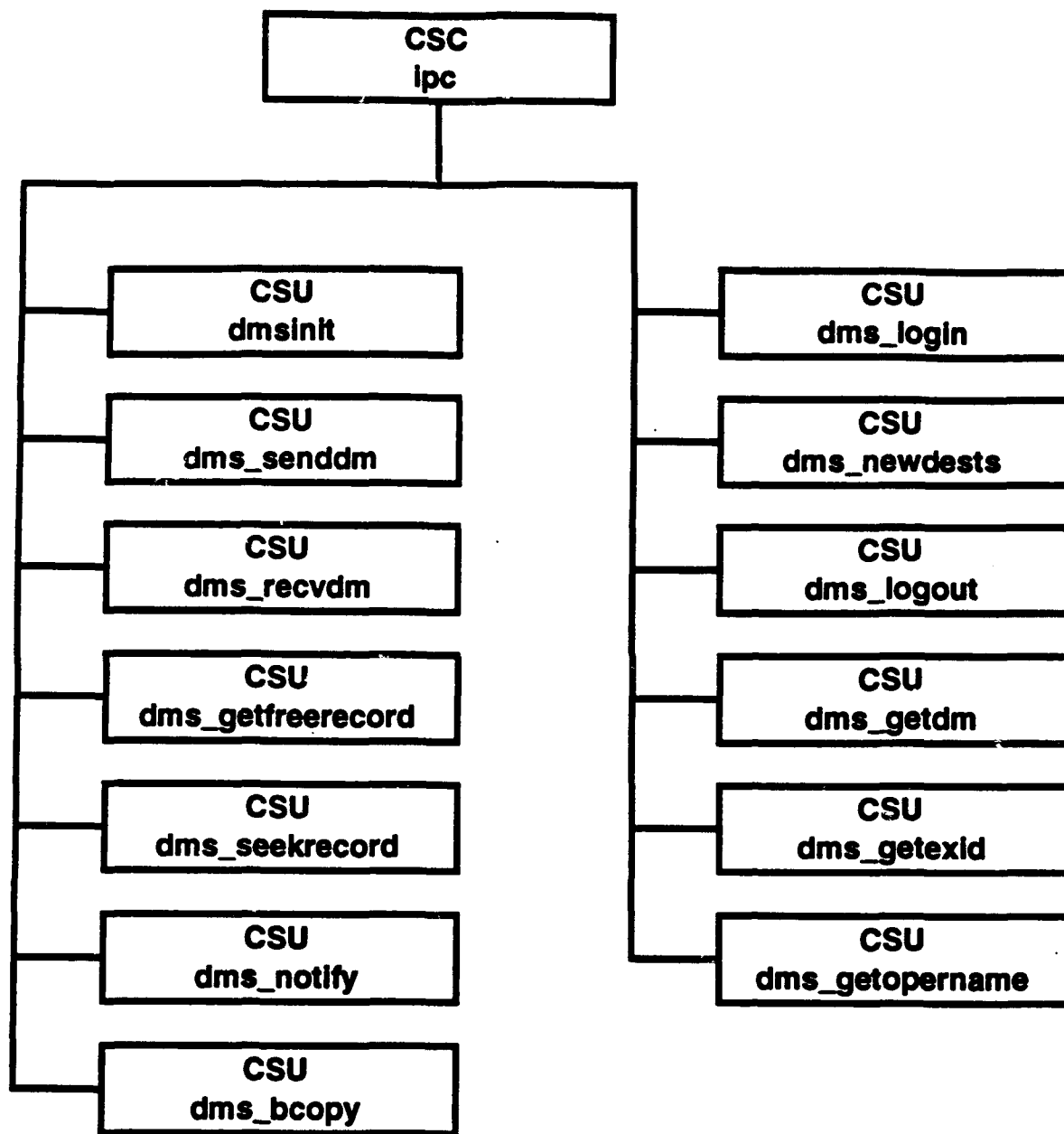


Figure 8 CSC ipc Structure

CSC ipc partially satisfies System Segment Specification requirements 3.2.1.2.1 and 3.2.1.2.2.1.

The following paragraphs under this section describe the relationship of the ipc CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the ipc CSC.

#### 4.3.1 CSU dmsinit.

The dmsinit CSU handles attaching to the message queues and to the shared memory segment. The following paragraphs provide design information for this CSU.

##### 4.3.1.1 CSU dmsinit requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.1.2 CSU dmsinit design.

The information identified below represents the detailed design of the dmsinit CSU.

a. Input/output data elements.

INPUTS:

"mode" indicates whether the caller is receiving or transmitting to the Ethernet

OUTPUTS: None

b. Local data elements.

"shmem\_size" is the maximum size of the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. returns 0 if unable to attach to queues and shared memory

f. Data conversion. None



- g. Use of other elements. Other elements that are used by the dmsinit CSU:  
"fanoutqid"  
"incoming\_shmid"  
"outgoingqid"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.2 CSU dms\_senddm.

The dms\_senddm puts a message in the Outgoing queue. The following paragraphs provide design information for this CSU.

##### 4.3.2.1 CSU dms\_senddm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.2.2 CSU dms\_senddm design.

The information identified below represents the detailed design of the dms\_senddm CSU.

- a. Input/output data elements.  
  
INPUTS:  
"buf" is a pointer to the message  
"buflen" indicates the size of the message  
  
OUTPUTS: None
- b. Local data elements.  
  
None
- c. Interrupts and signals. None

- d. Algorithms. None
- e. Error handling. returns 0 if unable to put the message in the queue
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the dms\_senddm CSU:  
"outgoingqid"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.3 CSU dms\_recvdm.

The dms\_recvdm waits for any message to appear in the Outgoing queue. The following paragraphs provide design information for this CSU.

##### 4.3.3.1 CSU dms\_recvdm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.3.2 CSU dms\_recvdm design.

The information identified below represents the detailed design of the dms\_recvdm CSU.

###### a. Input/output data elements.

###### INPUTS:

"buf" is a pointer to a buffer to contain the received message

###### OUTPUTS:

"buf" is a pointer to a buffer to contain the received message

- b. Local data elements.  
"msgin" contains the received message
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. Returns 0 if unable to retrieve the message from the queue
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the dms\_recvdm CSU:  
"outgoingqid"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.3.4 CSU dms\_getfreerecord.**

The dms\_getfreerecord CSU tries to find a free record in shared memory by calling dms\_seekrecord CSU, and on error, it will sleep and try again, for a limit of MAXRETRY. The following paragraphs provide design information for this CSU.

##### **4.3.4.1 CSU dms\_getfreerecord requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.3.4.2 CSU dms\_getfreerecord design.**

The information identified below represents the detailed design of the dms\_getfreerecord CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:

"dms\_getfreerecord" contains the address of a free record in shared memory

b. Local data elements.

"recaddr" contains address of free record

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Return 0 if unable to find a free record

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_getfreerecord CSU: None

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.3.5 CSU dms\_seekrecord.

The dms\_seekrecord CSU finds the next free record in the shared memory segment and returns the address of the record. The following paragraphs provide design information for this CSU.

##### 4.3.5.1 CSU dms\_seekrecord requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.3.5.2 CSU dms\_seekrecord design.

The information identified below represents the detailed design of the dms\_seekrecord CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:

"dms\_seekrecord" contains the address of a free record in shared memory

b. Local data elements.

"maxshmem" contains the highest address in the shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to find a free record

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_seekrecord CSU:  
"shmaddr"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.3.6 CSU dms\_notify.

The dms\_notify CSU sends a notification of a new message arrival to dms CSU. The following paragraphs provide design information for this CSU.

#### 4.3.6.1 CSU dms\_notify requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.3.6.2 CSU dms\_notify design.

The information identified below represents the detailed design of the dms\_notify CSU.

a. Input/output data elements.

INPUTS:

"new\_dm\_address" contains the address of the new message in the shared memory segment

OUTPUTS: None

b. Local data elements.

"msgout" contains the MSG\_DMAVAIL message

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to put the message in the queue

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_notify CSU:  
"fanoutqid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.3.7 CSU dms\_bcopy.

The dms\_bcopy CSU copies a buffer from the caller into shared memory, and then notifies dms CSU. The following paragraphs provide design information for this CSU.

##### 4.3.7.1 CSU dms\_bcopy requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.7.2 CSU dms\_bcopy design.

The information identified below represents the detailed design of the dms\_bcopy CSU.

a. Input/output data elements.

INPUTS:

"buf" is a pointer to a buffer containing the message

"buflen" contains the length of the buffer

"dests" is a pointer to a vector of CEOI strings

"exid" is byte containing the id of the exercise

OUTPUTS: None

b. Local data elements.

"nextrec" contains address of next free record in shared memory segment

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. Returns 0 if unable to copy new message into shared memory segment

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_bcopy CSU: None

h. Logic flow. Single pass.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.8 CSU dmslogin.

The dmslogin CSU creates a child process for the caller to handle communication with dms CSU. The child then sends a login message to dms CSU. The child process enters an infinite for loop waiting for a new message arrival notification from the dms CSU. The following paragraphs provide design information for this CSU.

##### 4.3.8.1 CSU dmslogin requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.8.2 CSU dmslogin design.

The information identified below represents the detailed design of the dmslogin CSU.

- a. Input/output data elements.

###### INPUTS:

"dests" is a pointer to a vector of CEOI strings  
"exid" contains the exercise id of the operator

###### OUTPUTS: None

- b. Local data elements.

"p" contains id of pipes  
"msgout" contains message sent out to fan-out queue  
"msgin" contains messages received from fan-out queue  
"shmem\_size" contains the size of the shared memory segment

- c. Interrupts and signals. None
- d. Algorithms. None



- e. Error handling. Returns 0 if unable to create pipes
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the dmslogin CSU:
  - "incoming\_shmem"
  - "fanoutqid"
  - "outgoingqid"
  - "childpid"
  - "cur\_exerciseid"
  - "cur\_opername"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.9 CSU dmsnewdests.

The dmsnewdests CSU sends a new set of CEOIs and Groups that the operator wishes to receive. The following paragraphs provide design information for this CSU.

##### 4.3.9.1 CSU dmsnewdests requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.9.2 CSU dmsnewdests design.

The information identified below represents the detailed design of the dmsnewdests CSU.

- a. Input/output data elements.

#### INPUTS:

"ceoi" contains the new vector of CEOI/Group strings

OUTPUTS: None

- b. Local data elements.  
"msgout" contains message sent out to fan-out queue
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. Returns 0 if unable to put message in queue
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the dmsnewdests CSU:  
"fanoutqid"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.10. CSU dmslogout.

The dmslogout CSU sends a logout message to the dms CSC. The following paragraphs provide design information for this CSU.

##### 4.3.10.1 CSU dmslogout requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.10.2 CSU dmslogout design.

The information identified below represents the detailed design of the dmslogout CSU.

- a. Input/output data elements.  
INPUTS: None  
OUTPUTS: None
- b. Local data elements.  
"msgout" contains message sent out to fan-out queue
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. Returns 0 if unable to put message in queue
- f. Data conversion. None
- g. Use of other elements. Other elements that are used by the dmslogout CSU:  
"fanoutqid"  
"childpid"
- h. Logic flow. Single pass.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.3.11 CSU dms\_getdm.

The dms\_getdm CSU reads from the pipe between the X-Client parent and its child. The CSU hangs until the read completes. The following paragraphs provide design information for this CSU.

##### 4.3.11.1. CSU dms\_getdm requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.3.11.2 CSU dms\_getdm design.**

The information identified below represents the detailed design of the dms\_getdm CSU.

a. Input/output data elements.

INPUTS:

"p" contains the id of the pipe

"pdu" is a pointer to buffer that will contain the message read from the pipe

OUTPUTS:

"pdu" is a pointer to buffer that will contain the message read from the pipe

b. Local data elements.

"nbytes" contains number of bytes read

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_getdm CSU:  
"p"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

**4.3.12 CSU dms\_getexid.**

The dms\_getexid CSU returns the exercise id of the session to the caller. The following paragraphs provide design information for this CSU.

**4.3.12.1 CSU dms\_getexid requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.3.12.2 CSU dms\_getexid design.**

The information identified below represents the detailed design of the dms\_getexid CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:

"dms\_getexid" contains the exercise id of the session

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_getexid CSU:  
"cur\_exerciseid"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.3.13 CSU dms\_getopername

The dms\_getopername CSU returns the operators name to the caller. The following paragraphs provide design information for this CSU.

##### 4.3.13.1 CSU dms\_getopername requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.3.13.2 CSU dms\_getopername design.

The information identified below represents the detailed design of the dms\_getopername CSU.

a. Input/output data elements.

INPUTS: None

OUTPUTS:

"dms\_getopername" contains the operators name

b. Local data elements.

None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements. Other elements that are used by the dms\_getopername CSU:  
"cur\_opername"

h. Logic flow. Single pass.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.4. CSC Ethernet.

The following paragraphs under this section describe the relationship of the Ethernet CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the Ethernet CSC.

CSC Ethernet partially satisfies System Segment Specification 3.2.1.2.2.2 and 3.2.1.2.2.1.

##### 4.4.1. Sub-Level CSC dmcc\_sim\_rx.c.

The following paragraphs under this section describe the relationship of the dmcc\_sim\_rx Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

The following diagram depicts the structure of the dmcc\_sim\_tx sublevel CSC.

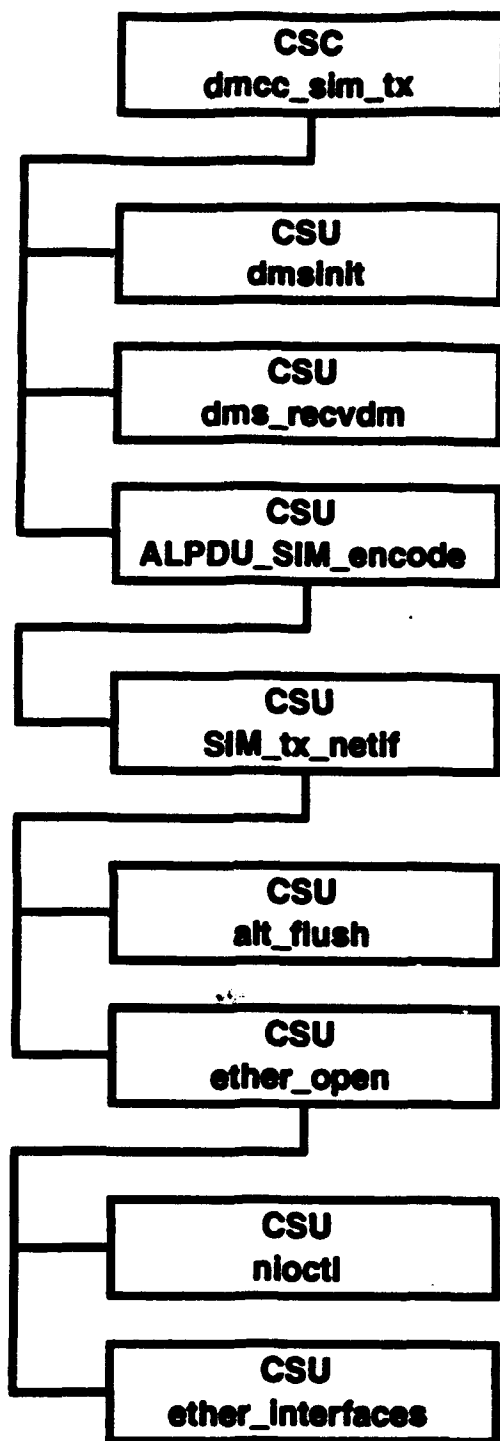


Figure 9 Sublevel CSU dmcc\_sim\_rx Structure



#### 4.4.1.1 CSU main

The main CSU is a driver for the `dmcc_sim_rx` function, and calls routines to attach shared memory, set ethernet interface mode for receiving packets, log message status counters, validate Ethernet type, validate the LLC sublayer header, and decode the input message all within a while-loop.

##### 4.4.1.1.1 CSU main requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

Syntax of invoking this CSU:

`dmcc_sim_rx`

a. Input/output data elements.

INPUTS:

"argv" is the command line input argument containing the Ethernet card name.

OUTPUTS:

None.

b. Local data elements.

"i" is an integer for-loop counter.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

display message with `printf`.

f. Data conversion. None

g. Use of other elements.

Data elements:

SIM\_interface\_name

Exercise\_ID

Library routines:

if - else

printf

exit

CSUs:

dmsinit

SIM\_rx\_netif

h. Logic flow.

set Exercise ID to pass all type of messages;

if command line argument is less than two

print the proper command usage;

else

get the SIMNET interface name from command line argument;

invoke dmsinit to initialize message queue;

if dmsinit return error

output error message and exit;

invoke SIM\_rx\_netif;

i. Data structures. Nonej. Local data files or database. Nonek. Limitations. None**4.4.1.2 CSU SIM\_rx\_netif**

The SIM\_rx\_netif CSU call function to set the receiving Ethernet interface mode then within an infinite loop, it reads in the input message, validates its Ethernet type, validates the LLC sublayer header, calls function to update the message status counters and to decode the input message.

**4.4.1.2.1 CSU SIM\_rx\_netif requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.4.1.2.2. CSU SIM\_rx\_netif design.**

The information identified below represents the detailed design of the SIM\_rx\_netif CSU.

Syntax of invoking this CSU:

SIM\_rx\_netif();

a. Input/output data elements.

INPUTS:

None.

OUTPUTS:

None.

b. Local data elements.

"i" is an integer for-loop counter.

"pType" is an unsigned short integer containing the input message's Ethernet type.

"frame\_seq" is long or 64 bit integer indicating the frame sequence.

"timervalues" is an instance of struct timeval.

"timerzone" is an instance of struct timezone.

"iov[]" is an instance of struct iovec.

"byte\_read" is an integer storage containing the number of message byte read.

"prom" is of type ether\_addr.

"protocol\_id" is a local global unsigned integer storage containing the protocol id of the received message.

"etherPacket" is a static ether\_packet type containing the input message structure.

c. Interrupts and signals. None

d. Algorithms. None.

e. Error handling.

1. invoke shutdown\_SIM\_rx to display messages.
2. return 1 for error.

f. Data conversion. None

g. Use of other elements.

## Data elements:

SIM\_sigrec,  
 SIM\_rx\_etherFd,  
 SIM\_interface\_name,  
 SIM\_rx\_index,  
 frame\_buff[][].

## Library routines:

if - else,  
 while - loop,  
 printf,  
 gettimeofday,  
 memcpy,  
 readv.

## CSUs:

ether\_open,  
 shutdown\_SIM\_rx,  
 ALPDU\_SIM\_decode,  
 logger.

h. Logic flow.

initialize SIM\_sigrec and frame\_seq to zero;  
 set timerzone.tz\_dsttime = DST\_NONE( with no specific time zone);

invoke ether\_open to set the assigned Ethernet card with promiscuous receive mode;

if ether\_open return error  
 invoke shutdown\_SIM\_rx to output error message and return with one;

for i < the max of MAX\_DATAGRAM +( 1 for discard data storage)  
 set etherPacket.pktbuf[i] points to the address of  
 frame\_buff[i][0];

initialize header buffer's length field to the number of possible MAC header length and base field points to the beginning of the input message;

## While - forever loop

if the number of non-processed input message < maximum datagram  
 is allowed in the input buffer  
 initialize data buffer's len field to  
 maximum  
 number of byte a message is allowed and  
 base field to the current index of message

```

input buffer;

invoke readv to read in message;
if number of byte read is <= zero
    invoke shutdown_SIM_rx to output the
    proper error message then return to
    dmcc_sim_rx with an error;
else
    invoke gettimeofday to stamp the
    incoming message;

    if the input message's Ethernet type field
    is not an valid SIMNET message
        invoke logger to log the Bad Ether
        Type error count;
    else
        if protocol_id is not valid;
            invoke logger to log the
            BAD_PROTOCOL_ID counter;
        else if DSAP is not valid
            invoke logger to log the BAD_DSAP counter;
        else if SSAP is not valid
            invoke logger to log the BAD_SSAP counter;
        else if control is not valid
            invoke logger to log the BAD_CONTROL
            counter;
        else
            invoke ALPDU_SIM_decode to decode the input
            message;
            increment the circular SIM_rx_index;
            increment SIM_sigrec to indicate the number of
            messages that have sent to ALPDU_SIM_decode but
            not yet been processed;
    else
        invoke readv to read in the message to the
        last datagram storage are in input buffer;
        invoke logger to log the Trash Received
        message counter;
return to caller;

```

- i. Data structures. None
- j. Local data files or database. None

k. Limitations.

1. Superuser privilege is required to set ethernet function.
2. Only accept message if the number of not yet processed message does not exceed the limit of the input buffer.

#### 4.4.1.3 CSU ALPDU\_SIM\_decode

The ALPDU\_SIM\_decode CSU split up the incoming user data into individual ALPDUs since there may be multiple ALPDUs in one user datagram and call function to validate the Exercise ID and validate the ALPDU format.

##### 4.4.1.3.1 CSU ALPDU\_SIM\_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.3.2 CSU ALPDU\_SIM\_decode design.

The information identified below represents the detailed design of the ALPDU\_SIM\_decode CSU.

Syntax of invoking this CSU:

ALPDU\_SIM\_decode(address of the input message, length of input message/packet, timestamp);

a. Input/output data elements.

**INPUTS:**

"assoc\_pkt" is an unsigned character pointer that points to the input message/packet.

"max\_length" is an unsigned short integer storage containing the input message/packet length.

"timervalues" is of struct timeval containing the input message/packet timestamp.

**OUTPUTS:** None.

b. Local data elements.

"assoc\_length" is a static unsigned short integer storage containing the association length;

"PDU\_kind" is a static unsigned character storage containing the pdu type

"next\_assoc\_pkt" is a static unsigned character pointer that points to the next association layer pdu to be split.

"first\_frame" is an unsigned character flag indicating the first association layer pdu of the input packet.

"valid\_frame" is a Boolean flag to indicate if an association layer pdu is in valid format.

c. Interrupts and signals. Noned. Algorithms. Nonee. Error handling. return with an error integer of 1.f. Data conversion. Noneg. Use of other elements.

Data elements:

SIM\_Assoc\_Layer.

Library routines:

if - else,  
do - while,  
if.

CSUs:

check\_exercise\_ID\_SIM,  
Rcv\_SIM\_APDU,  
check\_SIM\_ALPDU,  
logger.

h. Logic flow.

initialize valid\_frame to FALSE;

if SIM\_Assoc\_Layer is set to FALSE

if return of check\_exercise\_ID\_SIM is valid

invoke Rcv\_SIM\_APDU with the NEW\_ALPDU  
parameter;

else

invoke logger to increment the BAD\_EXERCISE\_ID  
counter;

else

set next\_assoc\_pkt points to assoc\_pkt;

invoke valid\_frame = check\_SIM\_ALPDU

to check for valid ALPDU format;

```

if valid_frame
  set first_frame = TRUE;
  do - while max_length > 0 and    check_SIM_ALPDU() of the
    next ALPDU is in valid format

    if PDU_kind is not equal to padding
      invoke check_exercise_ID_SIM;          if valid exercise
      ID;
      if this is the first frame
        invoke Rcv_SIM_APDU with          the NEW_ALPDU
        parameter;
        set first_frame to FALSE;
      else
        invoke Rcv_SIM_APDU with          the CONT_ALPDU
        parameter;
      else invalid exercise ID
        invoke logger to increment the
        BAD_EXERCISE_ID counter;

    subtract the last ALPDU length from the original length by
    max_length =    max_length - assoc_length;

    advance the next_assoc_pkt pointer by
    next_assoc_pkt = next_assoc_pkt +    assoc_length;

if the entire packet is valid where valid_frame is TRUE
  invoke Rcv_SIM_APDU to send the packet
  onto SIMNET by passing the END_ALPDU
  parameter;

return valid_frame status to caller.

```

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations.

The global variable SIM\_Assoc\_Layer must be set TRUE to perform the ALPDU split up.

#### 4.4.1.4 CSU check\_exercise\_ID\_SIM

The check\_exercise\_ID\_SIM CSU



#### 4.4.1.4.1 CSU check\_exercise\_ID\_SIM requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.4.1.4.2. CSU check\_exercise\_ID\_SIM design.

The information identified below represents the detailed design of the check\_exercise\_ID\_SIM CSU.

Syntax of invoking this CSU:

a. Input/output data elements.

INPUTS:

"new\_PDU" is an unsigned character pointer that points to the input association layer pdu.

OUTPUTS:

"valid\_frame" is a Boolean flag to indicate the status of the input pdu's exercise ID

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

pass\_all,  
Exercise\_ID.

Library routines:

if.

CSUs:

None.

h. Logic flow.

set valid\_frame = TRUE;

if pass\_all is FALSE and the exercise ID of new\_PDU does not equal to Exercise\_ID

set valid\_frame = FALSE;

return valid\_frame;

i. Data structures. Nonej. Local data files or database. Nonek. Limitations.**4.4.1.5 CSU check\_SIM\_ALPDU**

The check\_SIM\_ALPDU CSU checks for a valid ALPDU format. It returns FALSE if invalid and logs the error of TRUE if valid.

**4.4.1.5.1 CSU check\_SIM\_ALPDU design requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.4.1.5.2. CSU check\_SIM\_ALPDU design.**

The information identified below represents the detailed design of the check\_SIM\_ALPDU CSU.

Syntax of invoking this CSU:

check\_SIM\_ALPDU(assoc\_pkt, max\_length, assoc\_length, PDU\_kind);

a. Input/output data elements.**INPUTS:**

"assoc\_pkt" is an unsigned character pointer that points to the input association layer pdu address.

"max\_length" is an unsigned short integer containing the number of input packet bytes left for splitting.

"assoc\_length" is an unsigned short pointer that points to the number bytes in the input association layer pdu.

"PDU\_kind" is an unsigned character pointer that points to the association layer pdu's type.

#### OUTPUTS:

"valid\_frame" is an integer storage containing the Boolean value to be returned. It indicates whether or not the input association layer pdu is in a valid ALPDU format.

b. Local data elements.

"assoc\_PDU" is a pointer of AssocPDU\_type that points to the input association pdu.

c. Interrupts and signals. None

d. Algorithms.

set up association parameters.

check to see if length is correct and still bytes to process.

check to see if its a valid association pdu.

check to see if its a Digital Message Protocol PDU.

return the status of check.

e. Error handling.

invoke logger() to log all errors in a counter format.

f. Data conversion. None

g. Use of other elements.

Data elements:

ASSOC\_hdr\_leng.

Library routines:

if - else if

CSUs:

logger.

h. Logic flow.

initialize valid\_frame to TRUE;

initialize assoc\_PDU points to assoc\_pkt;

assign the input assoc\_PDU->PDUkind to PDU\_kind;

set    association    length    by    \*assoc\_length    =  
       ASSOC\_hdr\_leng[\*PDU\_kind] +  
 (assoc\_PDU->DataLength \* Assoc\_MULTIPLIER);

```
if (*assoc_length > max_length) or (max_length <= zero)
    set valid_frame to FALSE;
    invoke logger to increment the          BAD_ASSOC_LENGTH
    counter;

else if (*PDU_kind equals to an unknown kind) or (*PDU_kind > the
    possible association kind)
    set valid_frame to FALSE;
    invoke logger to increment the          BAD_ASSOC_KIND
    counter;

else if (assoc_PDU->Protocol does not equals to Digital Message
    Protocol PDU)
    set valid_frame to FALSE;
    invoke logger to increment the
    BAD_ASSOC_PROTOCOL counter;

return valid_frame to caller;

i.   Data structures. None
j.   Local data files or database. None
k.   Limitations. None
```

#### 4.4.1.6 CSU Rcv\_SIM\_ALPDU

The Rcv\_SIM\_ALPDU CSU strips out the incoming PDU and releases the receive buffer by decrementing the global variable sigrec when through processing that PDU and END\_ALPDU is received.

##### 4.4.1.6.1 CSU Rcv\_SIM\_ALPDU design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.6.2 CSU Rcv\_SIM\_ALPDU design.

The information identified below represents the detailed design of the Rcv\_SIM\_ALPDU CSU.

Syntax of invoking this CSU:

Rcv\_SIM\_ALPDU(end\_ALPDU, assoc\_pkt, assoc\_length,  
timervalues);

a. Input/output data elements.

INPUTS:

"end\_ALPDU" is an unsigned character containing the sequence of input association pdu.

"assoc\_pkt" is an unsigned character pointer that points to the input association layer pdu address.

"assoc\_length" is an unsigned short integer storage containing the number bytes in the input association layer pdu.

"timervalues" is of struct timeval containing the input packet timestamp.

OUTPUTS: None

b. Local data elements.

"assoc\_PDU" is a pointer of AssocPDU\_type that points to the input association pdu.

"ALPDU\_kind" is an unsigned character containing the input association pdu type.

"PDU\_length" is an unsigned short storage containing the input association pdu length in number of bytes.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

ASSOC\_hdr\_leng,

SIM\_Assoc\_Layer.

Library routines:

if

switch - case

CSUs:

decode\_SIM\_pkt.

h. Logic flow.

initialize ALPDU\_kind to an unknown kind;  
initialize assoc\_PDU to point to assoc\_pkt;  
initialize PDU\_length to assoc\_length;

if SIM\_Assoc\_Layer is TRUE

set ALPDU\_kind = assoc\_PDU->PDUKind;  
set PDU\_length less the  
ASSOC\_hdr\_leng[ALPDU\_kind];

switch (end\_ALPDU)

case continue ALPDU

invoke decode\_SIM\_pkt to decode the  
input pdu;

case end ALPDU

decrement the SIM\_sigrec to indicate the  
input packet is processed and is about to  
sent out to the net;

case new ALPDU

invoke decode\_SIM\_pkt to decode the  
input pdu;

return to caller;

i. Data structures. None

j. Local data files or database. None

k. Limitations.

This CSU depends on the global variable SIM\_Assoc\_Layer to know whether the Association Layer is present and should be processed or not.

This CSU assumes the calling routine has filtered out all Padding APDUs and APDUs which carry on data.

#### 4.4.1.7 CSU dms\_decode\_SIM\_pkt

The dms\_decode\_SIM\_pkt CSU checks for a valid ALPDU format. It returns FALSE if invalid and logs the error of TRUE if valid.

#### 4.4.1.7.1 CSU dms\_decode\_SIM\_pkt design requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.4.1.7.2. CSU dms\_decode\_SIM\_pkt design.

The information identified below represents the detailed design of the dms\_decode\_SIM\_pkt CSU.

Syntax of invoking this CSU:

```
dms_decode_SIM_pkt(time, old_pkt, new_pkt, max_length,  
ALPDU_kind);
```

a. Input/output data elements.

INPUTS:

"time" is of structure timeval containing the input association pdu timestamp - not used.

"old\_pkt" is an unsigned character points that points to the input pdu.

"new\_pkt" is an unsigned character points - not used(NULL is passed in).

"max\_length" is an unsigned short storage containing the number bytes in the input association layer pdu.

"ALPDU\_kind" is an unsigned character storage containing the type of pdu input - not used(NULL is passed in).

OUTPUTS: None.

b. Local data elements.

"pduhdr" is a pointer of DMC\_Header\_type that points to the input association pdu.

"pducmn" is a pointer of DMC\_CommonBlock\_type.

c. Interrupts and signals. None

d. Algorithms.

check the PDU to make sure it is a Digital Message else return to caller.  
Offset to CommonBlock to get the CEOs.

send the received datagram to "dms\_copy" to copy buffer to DMS Shared Memory.

e. Error handling.

output error message with printf if dms\_bcopy failed to copy buffer to DMS Shared Memory.

f. Data conversion. Noneg. Use of other elements.

Data elements:

ASSOC\_hdr\_leng.

Library routines:

sizeof

if

printf

CSUs:

dms\_bcopy

h. Logic flow.

set pduhdr points to old\_pkt;

if pduhdr->HeaderType.SIM.Header.kind is not a Digital Message  
return to caller;

offset to common block with pducmn = (DMC\_CommonBlock\_type \*)  
(old\_pkt + sizeof(DMC\_Header\_type);

invoke dms\_copy with old\_pkt, max\_length, pducmn->TargetCEOI,  
and pduhdr->HeaderType.SIM.Header.exercise as parameters;

if dms\_copy returns zero  
display error message " failed to copy buffer  
to DMS Shared Memory ";

return to caller;

i. Data structures. Nonej. Local data files or database. Nonek. Limitations. None**4.4.1.8 CSU logger**

The logger CSU logs any unrecognized data or error conditions that may occur while processing PDUs when logger is invoked.



#### 4.4.1.8.1 CSU logger design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.4.1.8.2 CSU logger design.

The information identified below represents the detailed design of the logger CSU.

Syntax of invoking this CSU:

logger(errcode, pkt\_add, pkt\_length);

a. Input/output data elements.

INPUTS:

"errcode" is an unsigned character storage containing the error value which indicating the type of error being logged.

"pkt\_add" is an unsigned character pointer that points to the address of the error pdu.

"pkt\_length" is an unsigned short storage containing the length of the error pdu in number of bytes.

OUTPUTS: None.

b. Local data elements.

"now" is of time\_t type.

"s[]" is a character array of size 100.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

output error message with printf.

f. Data conversion. None

g. Use of other elements.

Data elements:

"logger\_datafile" is an integer storage containing the logger file descriptor,

"FILE" file type.

**Library routines:**

fopen  
strftime  
fprintf  
if  
printf  
fflush

**CSUs:**

None

**h. Logic flow.**

if logger\_datafile is NULL

    invoke fopen to create "logger\_file.dat"  
    with "append" option;

if file descriptor returned by fopen equals to  
NULL

    printf error message "could not open  
    logger\_file.dat";  
    return to caller;

set now = time(NULL);

get current time by invoking strftime(s, 100,  
"\n%H:%M:%S on %A, %d %B %Y",  
localtime(&now));

invoke fprintf to write current time in s to  
logger\_datafile;

invoke fprintf to write filename to  
logger\_datafile;

invoke fprintf to write the message/error to  
logger\_datafile;

invoke fflush to flush the file descriptor;  
return to caller;

**i. Data structures. None****j. Local data files or database. None****k. Limitations. None**

#### 4.4.1.9 CSU ether\_open

The ether\_open CSU returns file descriptor for Ethernet device by name("ie0", "le0", etc...). It invoke niocli CSU to call the ioctl command for opening the Ethernet device.

##### 4.4.1.9.1 CSU ether\_open design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.9.2 CSU ether\_open design.

The information identified below represents the detailed design of the ether\_open CSU.

Syntax of invoking this CSU:

ether\_open( name, type, address);

a. Input/output data elements.

INPUTS:

"name" is a character pointer containing the Ethernet interface name.

"type" is an unsigned integer containing Ethernet packet type.

"address" is an ether\_addr type pointer containing the receiving Ethernet address.

OUTPUTS:

"fd" is an integer storage containing the file descriptor of the Ethernet to be returned.

b. Local data elements.

"ifr" is of ifreq structure type;

"filter" is of packetfilt type;

"fptr" is an unsigned character pointer;

"interfaces" is of char\*\*;

"i" is an integer register;

"flag" is a long or 64-bit integer flag;

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

None

Library routines:

if

perror

close

ioctl

strncpy

CSUs:

None

h. Logic flow.

if name is NULL get default ethernet interface

interface = ether\_interfaces();

if ether\_interface returns zero

return -1 to caller;

copy interface name: name = \*interface;

invoke open to open NIT\_DEV;

if return error

return -1 to caller;

if NIT\_DEV file descriptor >= FD\_SETSIZE

close the file descriptor with close;

set errno = empty file;

return -1 to caller;

invoke nioclt(fd, I\_SRDOPT, (char \*) RMSGD) to get discrete message  
from stream;

if return error

return -1 to caller;

if type does not equals to ETHER\_ALLTYPES

if type > ETHER\_ALLTYPES

close file descriptor with close;

set errno = EINVAL;

return -1 to caller;

```

/* push the filter to prevent being flooded with extraneous packets */
if (nioctl(fd, I_PUSH, "pf") < 0
    return -1 to caller;

fptr = &filter.Pf_Filter[0];
*fptr++ = ENF_PUSHPWORD + ETHER_TYPE / sizeof (short);
*fptr++ = ENF_PUSHLIT | ENF_EQ;
*fptr++ = htons ((u_short)type);

filter.Pf_FilterLen = fptr - &filter.Pf_Filter[0];
filter.Pf_Priority = 1;

if (nioctl(fd, NIOSETF, (char*)&filter) < 0)
    return -1 to caller;

/* binding */
invoke strncpy (ifr.ifr_name, name, sizeof(ifr.ifr_name));

if (nioctl(fd, NIOCBIND, (char*)&ifr) < 0)
    return -1 to caller;

if (addressx is not 0)
    if address->bytes[0] is not set to PROMISCUOUS_MODE
        if ether_cmp(address, &promiscuous)

            if MULTICAST is not a compilation      define    ether_addr
                local_addr;

            if not ETHER_MCAST (address)
                close the file descriptor;
                set errno = EINVAL;
                return -1 to caller;

if MULTICAST is a compilation define
    exit with -1 for error;
else
    if fptr equals to zero
        if (nioctl(fd, I_PUSH, "pf") < 0
            return -1 to caller;

    fptr = (unsigned short *)    &filter.Pf_Filter[0];
    if (type does not equal to    ETHER_ALLTYPES )
        *fptr++ = ENF_PUSHPWORD +    ETHER_TYPE /
        sizeof(short);
        *fptr++ = ENF+PUSHLIT;
        *fptr++ = htons( (u_short) type);

```

```
    *fptr++ = ENF_CAND;

/* compare the address */
for i < 3
    *fptr++ = (unsigned short)
        (ENF_PUSHPWORD + ETHER_DST) /
        sizeof (short) + i);
    *fptr++ = (unsigned short)
        (ENF_PUSHLIT | ENF_EQ);
    *fptr++ = (unsigned short)
        (address->shorts[i]);
    *fptr++ = (unsigned short)
        (ENF_PUSHPWORD + ETHER_DST /
        sizeof (short) + i;

    if SHOW_BCAST is a compilation
    define
        *fptr++ = (unsigned short)
            (ENF_PUSHLIT | ENF_EQ);
        *fptr++ = (unsigned short)
            (ether_bcast_addr.shorts[i]);
        *fptr++ = (unsigned short) ENF_OR;

    if SHOW_TCPIP is a compilation define
        *fptr++ = ENF_PUSHLIT | ENF_EQ;
    else
        *fptr++ = ENF_PUSHLIT | ENF_NEQ;

    filter.Pf_FilterLen = fptr -
        &filter.Pf_Filter[0];
    filter.Pf_Priority = 1;

    if (niocctl (fd, NIOCSETF, (char *) &filter) <
        0)
        return -1 to caller;

    if MULTICAST is a compilation define
    else

        flag = NI_PROMISC;
        if (niocctl(fd, NIOSCFLAGS, (char *) &flag
            < 0)
            return -1 to caller;

    else if in PROMISCUOUS_MODE
        flag = NI_PROMISC;
```

```
if (niocctl(fd, NIOCSFLAGS, (char*)&flag <
    0)
    return -1 to caller;
```

```
ether_type = type;
return file descriptor(fd) to caller;
```

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.4.1.10 CSU niocctl

The niocctl CSU invoke the ioctl function to set the input/output control.

##### 4.4.1.10.1 niocctl CSU design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.10.2 niocctl CSU design.

The information identified below represents the detailed design of the niocctl CSU.

Syntax of invoking this CSU:

niocctl (fd, cmd, ptr);

- a. Input/output data elements.

##### INPUTS:

"fd" is an integer storage containing the file descriptor of to set I/O control on.

"cmd" is an integer storage containing the I/O control command value.

"ptr" is a character pointer containing the address of a ifconf structure type.

OUTPUTS: None.

- b. Local data elements. None
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling.
- f. Data conversion. None
- g. Use of other elements.  
Data elements:  
    None  
Library routines:  
    ioctl  
    close  
CSUs:  
    None
- h. Logic flow.  
invoke ioctl with input fd, cmd, ptr;  
if ioctl return < 0  
    close fd;  
    return -1 to caller;  
return zero to caller;
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.4.1.11 CSU ether\_interfaces

The ether\_interfaces CSU returns an array of strings each entry of which is an ethernet interface name valid for use in ether\_open().



**4.4.1.11.1 CSU ether\_interfaces design requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.4.1.11.2 CSU ether\_interfaces design.**

The information identified below represents the detailed design of the ether\_interfaces CSU.

Syntax of invoking this CSU:

ether\_interface();

**a. Input/output data elements.**

INPUTS: None.

OUTPUTS:

"result" is a static character pointer that points to an array of size 16.

**b. Local data elements.**

/\* local global \*/

"checked" is a static boolean character flag;

"names" is a static character buffer of size 256.

"validname[]" is a static character array initialized with the first two possible interface name letter in a Sun workstation ("le", "ie", "ec", 0)

"\_ether\_ifindex[16]" is a short integer that is used by \_ether\_localif().

"\_ether\_ifconf" is fo struct ifconf.

"cbuf" is a static char buffer of size[16\*sizeof(struct ifreq);

/\* local data \*/

"sfd" is an integer storage containing the file descriptor.

"i,j,k,l" are integer registers.

"numinter" is an integer storage containing the number of interface.

**c. Interrupts and signals. None****d. Algorithms. None****e. Error handling. None**

f. Data conversion. None

g. Use of other elements.

Data elements:

None

Library routines:

close

if-else

goto

strcmp

strncmp

strncpy

for-loop

ioctl

sizeof

socket

if

CSUs:

None

h. Logic flow.

if checked is true

return result to caller;

open the socket by invoking `sfd = socket (AF_INET, sock_DGRAM, 0);`

if `sfd` is less than zero

return zero to caller;

`_ether_ifconf.ifc_len = sizeof (cbuf);`

`_ether_ifconf.ifc_buf = cbuf;`

if `(ioctl (sfd, SIOCGIFCONF, (char*)&ether_ifconf < 0)`

invoke close to close the file descriptor;

return zero to caller;

`k = 0;`

`numinter = _ether_ifconf.ifc_len / sizeof (struct ifreq);`

for `i < numinter`

for `j` not equals to zero

if `(strcmp i`

`(_ether_ifconf.ifc_req[i].ifr_name,`

`validname[j], 2) == 0)`

for `l < k`

if not `strcmp(result[l],`

`_ether_ifconf.ifc_req[i].ifr_name)`

`_ether_ifindex[l] = i;`

```
        goto duplicate;
        strncpy(name + 16 * k,
            _ether_ifconf.ifc_req[i].ifr_name,
            IFNAMSIZ);
        result[k] = names + 16 * k;
        if _ether_ifconf.ifc_req[i].ifr_addr.
            sa_family equals to AF_INET)
            _ether_ifindex[k] = i;
        else
            _ether_ifindex[k] = -1;
        ++k;
```

duplicate:

```
append zero to result;
close sfd;
set checked flag to one;
return interface name result to caller;
```

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.4.1.12 CSU shutdown\_SIM\_rx

The shutdown\_SIM\_rx CSU displays an error message based on the input shutdown code and close the SIM\_rx\_etherFd.

##### 4.4.1.12.1 CSU shutdown\_SIM\_rx design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.1.12.2 CSU shutdown\_SIM\_rx design.

The information identified below represents the detailed design of the shutdown\_SIM\_rx CSU.

Syntax of invoking this CSU:

```
shutdown_SIM_rx( shutdown_code);
```

a. Input/output data elements.

## INPUTS:

"shutdown\_code" is an integer value containing the shutdown code.

OUTPUTS: None.

b. Local data elements.c. Interrupts and signals. Noned. Algorithms. Nonee. Error handling.f. Data conversion. Noneg. Use of other elements.

Data elements:

SIM\_rx\_etherFd

Library routines:

perror

printf

CSUs:

None

h. Logic flow.

switch (shutdown\_code)

case 0 or 1

displays "Error opening stream:";

case 2

displays "can't receive data frame  
Ethernet";

message from

case 3

displays "Only LLC(SNAP) header is  
data"

read, no other

default

displays "invalid shutdown code"

close SIM\_rx\_etherFd;

return to caller;

i. Data structures. None

- j. Local data files or database. None
- k. Limitations. None

**4.4.1.13 CSU dmsinit**

refer to section 4.3.1

**4.4.1.14 CSU dms\_bcopy**

refer to section 4.3.7

**4.4.2 Sub-Level CSC dmcc\_sim\_tx.c.**

The following paragraphs under this section describe the relationship of the sim\_tx Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

The following diagram depicts the overall structure of the dmcc\_sim\_tx Sublevel CSC.

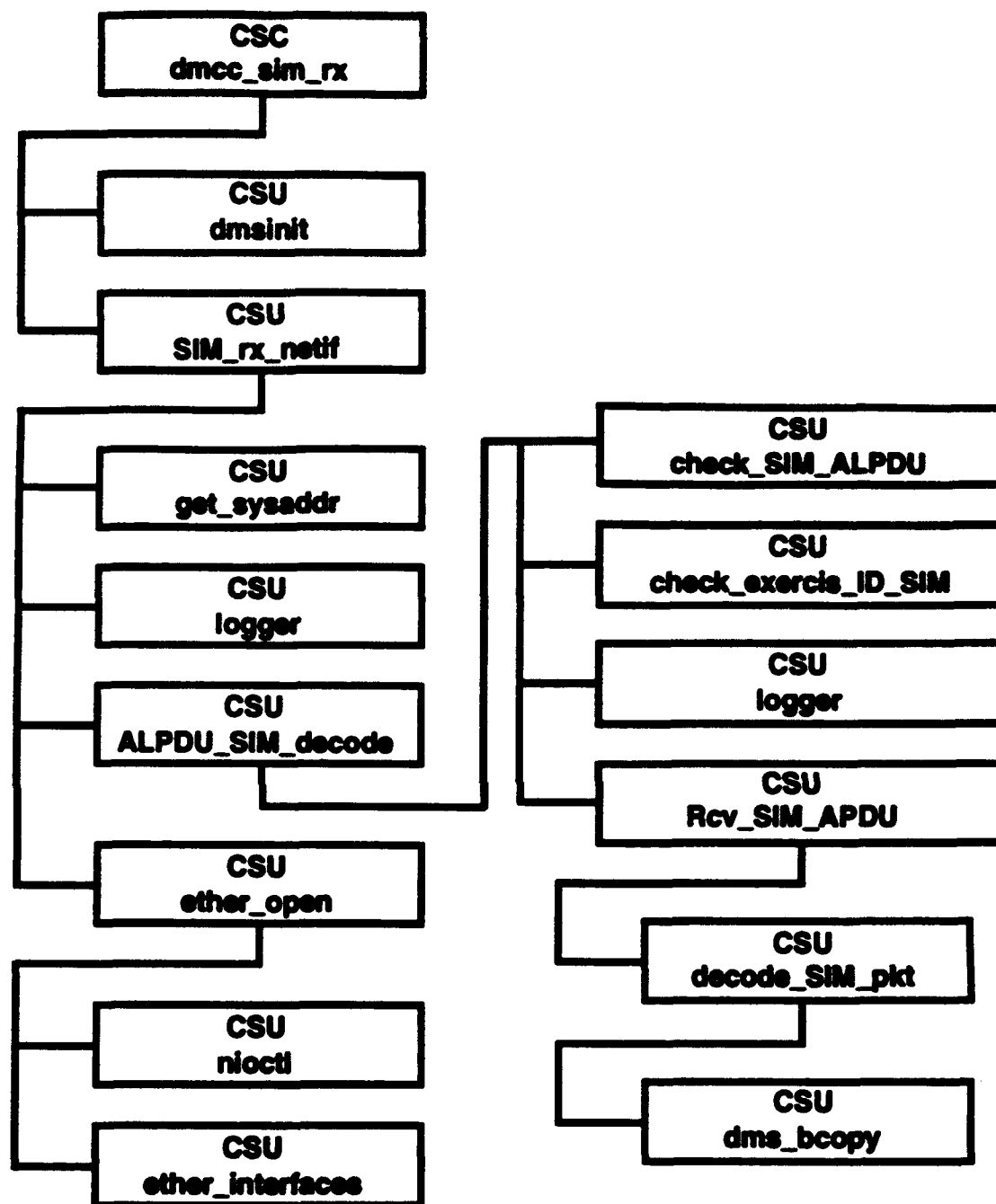


Figure 10 Sublevel CSU dmcc\_sim\_tx Structure

#### 4.4.2.1 CSU main

The main CSU is a driver for the dmcc\_sim\_tx function, gets Ethernet interface name from the command line argument, calls routines to initialize message queue, to gets pdu from message queue, and to send pdu to a message encode program.

##### 4.4.2.1.1 CSU main design requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.2.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

Syntax of invoking this CSU:

dmcc\_sim\_tx

a. Input/output data elements.

INPUTS:

"argv" is the command line input argument containing the Ethernet interface name.

OUTPUTS:

None.

b. Local data elements.

"i" is an integer for-loop counter.

"buff" is an unsigned character buffer of size 1500 for holding the input pdu.

"nbyte\_read" is an unsigned short integer storage containing the number of byte read in from the pdu.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

display message with printf.

f. Data conversion. None

g. Use of other elements.

Data elements:

SIM\_interface\_name

Library routines:

if - else

printf

bzero

exit

while-loop

CSUs:

dmsinit

dms\_recvdm

ALPDU\_SIM\_encode

h. Logic flow.

if command line argument is less than two

print the proper command usage;

else

get the SIMNET interface name from command line argument;

invoke dmsinit(

DMS\_ENET\_TRANSMIT) to

initialize message

queue;

if dmsinit return error

output error message and exit;

while forever

invoke bzero(buff, BUFF\_SIZE) to

initialize buff;

invoke nbyte\_read = dms\_recvdm to  
from message queue;

get input message

if nbyte\_read equals to zero

output error message and exit

with one;

invoke ALPDU\_SIM\_encode(

NEW\_ALPDU,

buff,

nbyte\_read) to encode the input pdu;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None



#### **4.4.2.2 CSU ALPDU\_SIM\_encode**

The ALPDU\_SIM\_encode CSU prepares a PDU for transmission by adding the Association Layer and sending the PDU to the network interface routine. The following paragraphs provide design information for this CSU.

##### **4.4.2.2.1 CSU ALPDU\_SIM\_encode requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.4.2.2.2 CSU ALPDU\_SIM\_encode design.**

The information identified below represents the detailed design of the ALPDU\_SIM\_encode CSU.

Syntax of invoking this CSU:

```
ALPDU_SIM_encode(  
    unsigned char end_ALPDU,  
    unsigned char *PDU_pkt,  
    unsigned short PDU_length);
```

a. Input/output data elements.

**INPUTS:**

"end\_ALPDU" is an unsigned character containing the sequence(type) of the input pdu\_pkt;

"pdu\_pkt" is a pointer to the first word of the buffer that is to be encoded and sent onto the network.

"pdu\_length" is a count of the number of bytes in the buffer.

**OUTPUTS:** None.

b. Local data elements.

"i" is a temporary counter used in various simple loops.

"padding" is a temporary counter used to in a loop to fill the transmit buffer with 0.

"offset" is a temporary variable used to position data in the transmit buffer.

"ALPDU\_kind" is an unsigned character storage containing the type of association pdu.

"assoc\_PDU" is an AssocPDU\_type pointer that points to SIM\_txbuf[][];

c. Interrupts and signals. None.

d. Algorithms. None.

e. Error handling. None.

f. Data conversion. None.

g. Use of other elements.

Data elements:

SIM\_tdbuf[]

SIM\_next\_frame

SIM\_Assoc\_Layer

Library routines:

if-else

for - loop

if

CSUs:

SIM\_tx\_netif

h. Logic flow.

if SIM\_Assoc\_Layer is TRUE

fill in association layer information

/\*add padding bytes if necessary\*/

if (padding = (PDU\_length % ASSOC\_MULTIPLIER)

padding = ASSOC\_MULTIPLIER -  
padding;

for i < padding

SIM\_tdbuf[SIM\_next\_frame][offset +  
PDU\_length + i] = 0x00;

set assoc\_PDU->DataLength = (unsigned char) ( (padding +  
PDU\_length) /  
ASSOC\_MULTIPLIER);

else

ALPDU\_kind =

ASSOC\_UNKNOWN\_KIND;

offset = SNAP\_LENGTH for  
fill in;

SIM\_tx\_netif() to

for i < PDU\_length

SIM\_tdbuf[SIM\_next\_frame][offset+i] = PDU\_pkt[i];

set the total pdu length: offset = offset +  
PDU\_length;

invoke SIM\_tx\_netif to send pdu to Ethernet with  
&SIM\_txbuf[SIM\_next\_frame[0], offset.

advance the circular input storage index:  
SIM\_next\_frame = (SIM\_next\_frame + 1) %  
MAX\_DATAGRAMS;

return to caller;

- i. Data structures. None.
- j. Local data files or database. None.
- k. Limitations. None.

#### 4.4.2.3 CSU SIM\_tx\_netif

The SIM\_tx\_netif CSU allows association Protocol Data Units (APDUs) to be transmitted onto the EtherNet as broadcast messages. The calling function specifies the start address and length of buffer to be sent to the Ethernet. The following paragraphs provide design information for this CSU.

##### 4.4.2.3.1 CSU SIM\_tx\_netif requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.2.3.2 CSU SIM\_tx\_netif design.

The information identified below represents the detailed design of the SIM\_tx\_netif CSU.

- a. Input/output data elements.

###### INPUTS:

"pdu\_in" is an unsigned character pointer to the first word of the buffer that is sent onto the network.

"pdu\_len" is an unsigned short containing the number of bytes in the input data.

OUTPUTS: None.

b. Local data elements.

"i,j" are integer loop counter.

"padding" is an integer storage containing the number of bytes to be padded with zero prior to transmission.

"data\_len" is an integer storage containing the number of padding bytes when multiply it by eight.

"assoc\_data\_len" is an integer storage containing the number of padding bytes.

"etherPacket" is a static ether\_packet storage.

"sa" is a static sockaddr structure.

"cbuf" is a static strbuf structure containing the pdu control information.

"dbuf" is a static strbuf structure containing the pdu information.

c. Interrupts and signals. None.

d. Algorithms. None.

e. Error handling. On transmission error, log the error and notify the operator.

f. Data conversion.

Data elements:

SIM\_tx\_etherFd

SIM\_interface\_name

SIM\_Assoc\_Layer

SIM\_send\_frame

Library routines:

if

perror

for-loop

if - else

sizeof

bcopy

CSUs:

ether\_open

logger

putmsg

g. Use of other elements. Other elements that are used by the SIM\_tx\_netif CSU: ether\_open, logger.

h. Logic flow. Simple IF ... Then ... Else, Simple for loop.

```

if (init_SIM_tx_netif is FALSE)
    invoke ether_open(SIM_interface_name,
                     ETHER_ALLTYPES,
                     NULL) to initialize the
    Ethernet interface;
if ether_open returns <= zero
    invoke perror to output error message;
    invoke exit with one;

set buffer fields for sending Ethernet frame:
sa.sa_family = AF_UNSPEC;
cbuf.maxlen = cbuf.len = sizeof(sa);
cbuf.buf = (char *)&sa;

set init_SIM_tx_netif to TRUE;

set the IEEE 802.3 LLC sublayer frame control header:
ALPDU_in[0] = AA;           - DSAP
ALPDU_in[1] = AA;           - SSAP
ALPDU_in[2] = 0x03;         - CNTL
ALPDU_in[3] = 0x080008;     - Protocol ID
ALPDU_in[6] = 0x5208;       - Ether Type

if (padding = ASSOC_PDU_MIN_LEN -
    ALPDU_len) > 0 ) padding requires
    if SIM_Assoc_Layer is TRUE
        append padding association layer at the end of the input pdu:
        1st byte of association layer =
            ASSOC_VER_PADKIND;
        2nd byte of association layer =
            padding/8;
        3rd byte of association layer =
            ASSOC_BROADCAST;
        4th byte of association layer =
            ASSOC_SIMULATION;
        5th to 8th byte of association layer =
            0x00;
    else
        pad the association layer with 0x00;

for j < # of padding bytes
    pad with 0x00;

ALPDU_len = ALPDU_len + association layer's 8 bytes + padding bytes;

etherPacket.LLClen = ALPDU_len;

```

```
etherPacket.pktlen = ALPDU_len;  
etherPacket.pktbuf[0] = ALPDU_in;
```

invoke bcopy((char\*)&etherPacket, sa.sa\_data, ETHER\_HDR\_SIZE) to copy the Ethernet header onto the control buffer;

```
set data buffer length to be sent:  
dbuf.len = etherPacket.pktlen;  
dbuf.buf = (char *)&etherPacket.pktbuf[0][0];
```

invoke putmsg(SIM\_tx\_etherFd, &dbuf, &dbuf, 0) to send Ethernet frame to STREAM;

```
if putmsg returns less than zero  
    invoke logger to log "PDU_BAD_SENT";  
    invoke perror to output error message;
```

signal ALPDU\_SIM\_encode that a frame is sent by decrementing SIM\_send\_frame;

return to caller;

i. Data structures. None.

j. Local data files or database. None.

k. Limitations.

FUNC must be defined at compilation time.

MODULE\_TEST must not be defined at compilation time.

#### 4.4.2.4 CSU dmsinit

refer to section 4.3.1

#### 4.4.2.5 CSU dms\_recvdm

refer to section 4.3.3

#### 4.4.2.6 CSU alt\_pause

The alt\_pause CSU stops the current process until a key is struck from the keyboard.

#### 4.4.2.6.1 CSU pause requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.4.2.6.2 CSU pause design.

The information identified below represents the detailed design of the pause CSU.

Syntax of invoking this CSU:

alt\_pause();

a. Input/output data elements.

INPUTS: None

OUTPUTS: None

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling.

f. Data conversion. None

g. Use of other elements.

Data elements:

stdin

Library routines:

printf

alt\_flush

fflush

getchar

CSUs:

None

h. Logic flow.

if #ifdef ALT\_FLUSH is defined as a compiler option  
invoke alt\_flush(stdin) to flush the standard input;

else invoke fflush(stdin) to flush the standard input;

invoke printf to output the "pause" message;  
invoke getchar;

if #ifdef ALT\_FLUSH is defined as a compiler option  
invoke alt\_flush(stdin) to flush the standard input;  
else invoke fflush(stdin) to flush the standard input;

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.4.2.7 CSU alt\_flush

The alt\_flush CSU flush all typeheads in the input buffer. alt\_flush CSU is used only if #ifdef ALT\_FLUSH is defined as a compiler option.

##### 4.4.2.7.1 CSU alt\_flush requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.4.2.7.2 CSU alt\_flush design.

The information identified below represents the detailed design of the alt\_flush CSU.

Syntax of invoking this CSU:  
alt\_flush(fp);

- a. Input/output data elements.

INPUTS:  
"fp" FILE pointer;

OUTPUTS:  
return 1.

- b. Local data elements.



c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

None

Library routines:

getc

fdopen

fclose

CSUs:

None

h. Logic flow.

```
while fp->cnt > zero
    invoke getc(fp);
return(1);
```

if the compiler control line

#if constant-expression evaluates to zero

declare fp\_tmp as FILE \*;

fp\_tmp = fdopen(fileno(fp), "r");

invoke fclose(fp) to close the original fp and  
lose its buffer;

set the original fp = fp\_tmp;

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.4.2.8 CSU logger

reference to 4.4.1.8

**4.4.2.9 CSU ether\_open**

reference to 4.4.1.9

**4.4.2.10 CSU niocli**

reference to 4.4.1.10

**4.4.2.11 CSU ether\_interfaces**

reference to 4.4.1.11

**4.5 CSC client**

The DMCC Client CSC is an event driven, X-windows application. It uses the OSF Motif® graphical user interface. It contains hand-coded C language routines as well as routines generated by a graphical user interface design environment (GUIDE) called Builders Xcessory, which is published by Integrated Computer Solutions, Incorporated.

The client CSC manages the X-Windows/OSF Motif® graphical user interface. The following diagram details the top level structure of this CSC.

The client CSC contains the functionalities which satisfy, in large measure, the actual DMCC SYstem Segment Specification requirements 3.2.1.2.2.2, d.2.1.2.2.3, 3.2.1.2.2.3.1, 3.2.1.2.2.3.2, 3.2.1.2.2.3.3, 3.2.1.2.2.3.4 and 3.2.1.2.2.3.5 and 3.2.1.2.2.3.6.

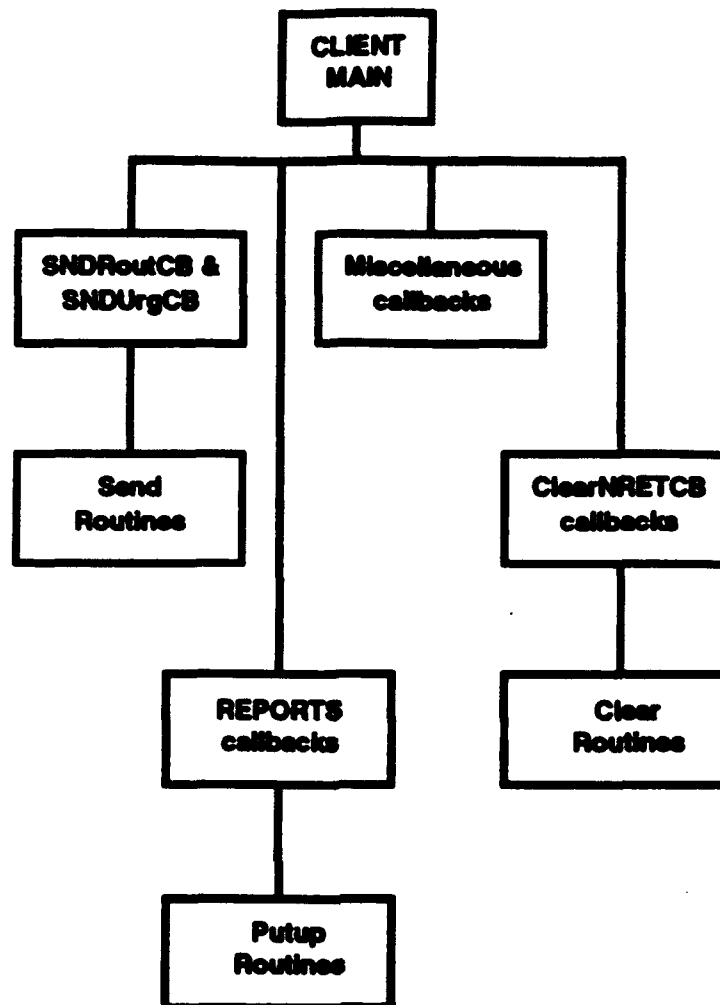


Figure 11 CSC Client Structure Chart

#### 4.5.1 Sub-Level CSC addrListCB.

The following paragraphs under this section describe the relationship of the addrListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.1.1 CSU AddrListAddCB

The AddrListAddCB CSU contains the callback for the address list entry screen add button. It takes the entry from the text field and adds it to the address list.

**4.5.1.1.1 CSU AddrListAddCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.1.1.2 CSU AddrListAddCB design.**

The information identified below represents the detailed design of the AddrListAddCB CSU.

**4.5.1.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.1.1.2.2 Local data elements.**

"text" - contains the text of the highlighted item

"numItems" - contains the number of items in the list

**4.5.1.1.2.3 Interrupts and signals. None****4.5.1.1.2.4 Algorithms. None****4.5.1.1.2.5 Error handling. None****4.5.1.1.2.6 Data conversion. None****4.5.1.1.2.7 Use of other elements. Other elements that are used by the AddrListAddCB CSU:**

"addrListList" - widget id of the address list

"addrListNewItem" - widget id of the new address text entry field

"addrListAddBtn" - widget id for the add button on the address list screen

"addrListDeleteBtn" - widget id for the delete button on the address list screen

"addressList" - array with all of the entries in the address list

**4.5.1.1.2.8 Logic flow. Single pass.**

4.5.1.1.2.9 Data structures. None

4.5.1.1.2.10 Local data files or database. None

4.5.1.1.2.11 Limitations. None

#### 4.5.1.2 CSU AddrListDeleteCB

The AddrListDeleteCB CSU contains the callback for the address list entry screen delete button. It takes the entry that is highlighted and deletes it from the address list.

##### 4.5.1.2.1 CSU AddrListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.1.2.2 CSU AddrListDeleteCB design.

The information identified below represents the detailed design of the AddrListDeleteCB CSU.

###### 4.5.1.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS:

###### 4.5.1.2.2.2 Local data elements.

"i" - counter for loop

"itemNum" - the item number from the list to be deleted

"numItems" - the number of items in the list

4.5.1.2.2.3 Interrupts and signals. None

4.5.1.2.2.4 Algorithms. None

4.5.1.2.2.5 Error handling. None

4.5.1.2.2.6 Data conversion. None

4.5.1.2.2.7 Use of other elements. Other elements that are used by the AddrListDeleteCB CSU:

"addrListList" - widget id of the address list

"addrListAddBtn" - widget id for the add button on the address list screen

"addrListDeleteBtn" - widget id for the delete button on the address list screen

"addressList" - array with all of the entries in the address list

4.5.1.2.2.8 Logic flow. Single pass.

4.5.1.2.2.9 Data structures. None

4.5.1.2.2.10 Local data files or database. None

4.5.1.2.2.11 Limitations. None

#### 4.5.1.3 CSU AddrListSaveExitCB

The AddrListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Address List screens and remanages the SysMain screens.

##### 4.5.1.3.1 CSU AddrListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.1.3.2 CSU AddrListSaveExitCB design.

The information identified below represents the detailed design of the AddrListSaveExitCB CSU.

###### 4.5.1.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS:

- 4.5.1.3.2.2 Local data elements. None
- 4.5.1.3.2.3 Interrupts and signals. None
- 4.5.1.3.2.4 Algorithms. None
- 4.5.1.3.2.5 Error handling. None
- 4.5.1.3.2.6 Data conversion. None
- 4.5.1.3.2.7 Use of other elements. Other elements that are used by the AddrListSaveExitCB CSU:
- 4.5.1.3.2.8 Logic flow. Single pass.
- 4.5.1.3.2.9 Data structures. None
- 4.5.1.3.2.10 Local data files or database. None
- 4.5.1.3.2.11 Limitations. None

#### **4.5.1.4 CSU AddrListClearExitCB**

The AddrListClearExitCB CSU handles the callback from the Clear and Exit pushbutton. It unmanages the Address List screens and remanages the SysMain screens.

##### **4.5.1.4.1 CSU AddrListClearExitCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.1.4.2 CSU AddrListClearExitCB design.**

The information identified below represents the detailed design of the AddrListClearExitCB CSU.

###### **4.5.1.4.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:**

- 4.5.1.4.2.2 Local data elements. None
- 4.5.1.4.2.3 Interrupts and signals. None
- 4.5.1.4.2.4 Algorithms. None
- 4.5.1.4.2.5 Error handling. None
- 4.5.1.4.2.6 Data conversion. None
- 4.5.1.4.2.7 Use of other elements. Other elements that are used by the AddrListClearExitCB CSU:
- 4.5.1.4.2.8 Logic flow. Single pass.
- 4.5.1.4.2.9 Data structures. None
- 4.5.1.4.2.10 Local data files or database. None
- 4.5.1.4.2.11 Limitations. None

**4.5.1.5 CSU PutupAddrList**

The PutupAddrList CSU creates the Address List entry screen if necessary and manages the objects with the first item of the list highlighted.

**4.5.1.5.1 CSU PutupAddrList requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.1.5.2 CSU PutupAddrList design.**

The information identified below represents the detailed design of the main CSU.

**4.5.1.5.2.1 Input/output data elements.****INPUTS:**



**OUTPUTS:**

- 4.5.1.5.2.2 Local data elements. None
- 4.5.1.5.2.3 Interrupts and signals. None
- 4.5.1.5.2.4 Algorithms. None
- 4.5.1.5.2.5 Error handling. None
- 4.5.1.5.2.6 Data conversion.
- 4.5.1.5.2.7 Use of other elements. Other elements that are used by the PutupAddrList CSU:
  - "addrListForm" - widget id for the address list entry form
  - "addrListTmi" - widget id for the address list TMI form
  - "addrListList" - widget id of the address list
- 4.5.1.5.2.8 Logic flow. Single pass.
- 4.5.1.5.2.9 Data structures. None
- 4.5.1.5.2.10 Local data files or database. None
- 4.5.1.5.2.11 Limitations. None

**4.5.2 Sub-Level CSC consoleCB.**

The following paragraphs under this section describe the relationship of the consoleCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.2.1 CSU ConsoleSysMainBtnCB**

The ConsoleSysMainBtnCB CSU handles the callback to from the SysMain pushbutton. It returns dmcc to the SysMain screen at any point during a session.

**4.5.2.1.1 CSU ConsoleSysMainBtnCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.2.1.2 CSU ConsoleSysMainBtnCB design.**

The information identified below represents the detailed design of the ConsoleSysMainBtnCB CSU.

**4.5.2.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:****4.5.2.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.2.1.2.3 Interrupts and signals. None****4.5.2.1.2.4 Algorithms. None****4.5.2.1.2.5 Error handling. None****4.5.2.1.2.6 Data conversion. None****4.5.2.1.2.7 Use of other elements. Other elements that are used by the ConsoleSysMainBtnCB CSU:**

"tmiScreen" - widget id which contains the id of the TMI form currently being displayed

"smdScreen" - widget id which contains the id of the entry form currently being displayed

"cikLabel2" - label of the cik2 label

**4.5.2.1.2.8 Logic flow. Single pass.****4.5.2.1.2.9 Data structures. None****4.5.2.1.2.10 Local data files or database. None****4.5.2.1.2.11 Limitations. None**

#### 4.5.3 Sub-Level CSC freeTxtCB.

The following paragraphs under this section describe the relationship of the freeTxtCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.3.1 CSU FreeTxtSendCB

The FreeTxtSendCB CSU handles the callback from both the Send Urgent and the Send Routine pushbuttons. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldFreeText to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

##### 4.5.3.1.1 CSU FreeTxtSendCB requirements.

The FreeTxtSendCB CSU satisfies section 3.2.1.2.2.2.2, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

##### 4.5.3.1.2 CSU FreeTxtSendCB design.

The information identified below represents the detailed design of the FreeTxtSendCB CSU.

##### 4.5.3.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS:

##### 4.5.3.1.2.2 Local data elements.

"targetCEOI" - the name of the target for the Digital Message

"text" - text from the free text window

"optText" - annotation text

"pri" - priority of the message

"nullChar" null character

##### 4.5.3.1.2.3 Interrupts and signals. None

##### 4.5.3.1.2.4 Algorithms. None

4.5.3.1.2.5 Error handling. None

4.5.3.1.2.6 Data conversion. None

4.5.3.1.2.7 Use of other elements. Other elements that are used by the FreeTxtSendCB CSU:

"FreeTxtAdd" - information on the FreeText address selection menu

"replyFlag" - boolean; is this a reply

"reuseFlag" - boolean; is this a reuse and include

"replyCEOI" - target name for reply call

"reusePDU" - PDU to be forwarded

4.5.3.1.2.8 Logic flow. Single pass.

4.5.3.1.2.9 Data structures. None

4.5.3.1.2.10 Local data files or database. None

4.5.3.1.2.11 Limitations. None

#### 4.5.3.2 CSU FreeTxtSaveReturn

The FreeTxtSaveReturn CSU handles the callback from the Save and Return pushbutton. It will unmanage the Free Text window and remanage the Reports window.

##### 4.5.3.2.1 CSU FreeTxtSaveReturn requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.3.2.2 CSU FreeTxtSaveReturn design.

The information identified below represents the detailed design of the FreeTxtSaveReturn CSU.

###### 4.5.3.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

- 4.5.3.2.2.2 Local data elements. None
- 4.5.3.2.2.3 Interrupts and signals. None
- 4.5.3.2.2.4 Algorithms. None
- 4.5.3.2.2.5 Error handling. None
- 4.5.3.2.2.6 Data conversion. None
- 4.5.3.2.2.7 Use of other elements. Other elements that are used by the FreeTxtSaveReturn CSU:
  - "freeTxtTmi" - widget id for Free Text TMI form
  - "freeTxtForm" - widget id for Free Text entry form
- 4.5.3.2.2.8 Logic flow. Single pass.
- 4.5.3.2.2.9 Data structures. None
- 4.5.3.2.2.10 Local data files or database. None
- 4.5.3.2.2.11 Limitations. None

**4.5.3.3 CSU FreeTxtClearReturn**

The FreeTxtClearReturn CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Free Text window and remanage the Reports window.

**4.5.3.3.1 CSU FreeTxtClearReturn requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.3.3.2 CSU FreeTxtClearReturn design.**

The information identified below represents the detailed design of the FreeTxtClearReturn CSU.

**4.5.3.3.2.1    Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.3.3.2.2    Local data elements. None**

**4.5.3.3.2.3    Interrupts and signals. None**

**4.5.3.3.2.4    Algorithms. None**

**4.5.3.3.2.5    Error handling. None**

**4.5.3.3.2.6    Data conversion. None**

**4.5.3.3.2.7    Use of other elements. Other elements that are used by  
the FreeTxtClearReturn CSU:**

"freeTxtTmi" - widget id for Free Text TMI form

"freeTxtForm" - widget id for Free Text entry form

**4.5.3.3.2.8    Logic flow. Single pass.**

**4.5.3.3.2.9    Data structures. None**

**4.5.3.3.2.10   Local data files or database. None**

**4.5.3.3.2.11   Limitations. None**

**4.5.3.4        CSU freetxtad\_moreCB**

The freetxtad\_moreCB CSU handles the callback from the address more button. It will only function if there are more than 4 addresses, in which case it will show a second bank of addresses.

**4.5.3.4.1      CSU freetxtad\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.5.3.4.2 CSU freetxtad\_moreCB design.

The information identified below represents the detailed design of the freetxtad\_moreCB CSU.

##### 4.5.3.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.3.4.2.2 Local data elements. None

4.5.3.4.2.3 Interrupts and signals. None

4.5.3.4.2.4 Algorithms. None

4.5.3.4.2.5 Error handling. None

4.5.3.4.2.6 Data conversion. None

4.5.3.4.2.7 Use of other elements. Other elements that are used by the freetxtad\_moreCB CSU:

"FreeTxtAdd" - information on the FreeText address selection menu

4.5.3.4.2.8 Logic flow. Single pass.

4.5.3.4.2.9 Data structures. None

4.5.3.4.2.10 Local data files or database. None

4.5.3.4.2.11 Limitations. None

#### 4.5.3.5 CSU freetxtad\_rotateCB

The freetxtad\_rotateCB CSU handles the callback from the address rotate button. It will only work if there is more than one address currently being displayed, in which case it will highlight the next address in the list or the first in the list if the last is currently highlighted.

#### 4.5.3.5.1 CSU freetxtad\_rotateCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.5.3.5.2 CSU freetxtad\_rotateCB design.

The information identified below represents the detailed design of the freetxtad\_rotateCB CSU.

##### 4.5.3.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.3.5.2.2 Local data elements. None

4.5.3.5.2.3 Interrupts and signals. None

4.5.3.5.2.4 Algorithms. None

4.5.3.5.2.5 Error handling. None

4.5.3.5.2.6 Data conversion. None

4.5.3.5.2.7 Use of other elements. Other elements that are used by the freetxtad\_rotateCB CSU:

"FreeTxtAdd" - information on the FreeText address selection menu

4.5.3.5.2.8 Logic flow. Single pass.

4.5.3.5.2.9 Data structures. None

4.5.3.5.2.10 Local data files or database. None

4.5.3.5.2.11 Limitations. None



#### **4.5.3.6 CSU PutupFreeTxt**

The PutupFreeTxt CSU will create the Free Text report screen if necessary and will also manage the screen.

##### **4.5.3.6.1 CSU PutupFreeTxt requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.3.6.2 CSU PutupFreeTxt design.**

The information identified below represents the detailed design of the PutupFreeTxt CSU.

###### **4.5.3.6.2.1 Input/output data elements.**

INPUTS:

OUTPUTS: .

###### **4.5.3.6.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"i" - counter

"entries" - number of address entries

###### **4.5.3.6.2.3 Interrupts and signals. None**

###### **4.5.3.6.2.4 Algorithms. None**

###### **4.5.3.6.2.5 Error handling. None**

###### **4.5.3.6.2.6 Data conversion. None**

###### **4.5.3.6.2.7 Use of other elements. Other elements that are used by the PutupFreeTxt CSU:**

"FreeTxtAdd" - information on the FreeText address selection menu

"freeTxtForm" - widget id for the free text entry form

"addressList" - list of addresses

4.5.3.6.2.8 Logic flow. Single pass.

4.5.3.6.2.9 Data structures. None

4.5.3.6.2.10 Local data files or database. None

4.5.3.6.2.11 Limitations. None

#### 4.5.3.7 CSU ClearFreeTxt

The ClearFreeTxt CSU will clear all entries selected in the Free Text report.

##### 4.5.3.7.1 CSU ClearFreeTxt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.3.7.2 CSU ClearFreeTxt design.

The information identified below represents the detailed design of the ClearFreeTxt CSU.

###### 4.5.3.7.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.3.7.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

###### 4.5.3.7.2.3 Interrupts and signals. None

###### 4.5.3.7.2.4 Algorithms. None

###### 4.5.3.7.2.5 Error handling. None

###### 4.5.3.7.2.6 Data conversion. None

4.5.3.7.2.7 Use of other elements. Other elements that are used by the ClearFreeTxt CSU:

"FreeTxtAdd" - information on the FreeText address selection menu

"textFreeTxt" - widget id of text area of Free Text screen

4.5.3.7.2.8 Logic flow. Single pass.

4.5.3.7.2.9 Data structures. None

4.5.3.7.2.10 Local data files or database. None

4.5.3.7.2.11 Limitations. None

#### 4.5.4 Sub-Level CSC getMessageCB.

The following paragraphs under this section describe the relationship of the getMessageCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.4.1 CSU getIncomingMsg

The getIncomingMsg CSU handles the receipt of any incoming message pdu. It is called when there is a message waiting to be read. It then reads the message from a pipe and processes it accordingly. If it is an acknowledgment, it will display a popup message with an ack message. If the queue is already full, it will replace the oldest routine message or oldest urgent if no routines exist. Otherwise, it will just place it in the queue and update the Msgs message list. It also updates the numMessages and the numUnread globals.

##### 4.5.4.1.1 CSU getIncomingMsg requirements.

The getIncomingMsg CSU satisfies section 3.2.1.2.2.1.2, 3.2.1.2.2.1.6; 3.2.1.2.2.3.4.1 and 3.2.1.2.2.1.6.3 of the specific requirements of the DMCC system.

##### 4.5.4.1.2 CSU getIncomingMsg design.

The information identified below represents the detailed design of the getIncomingMsg CSU.

##### 4.5.4.1.2.1 Input/output data elements.

###### INPUTS:

"cli\_data" - client data

"sourceFd" - pointer to a pipe file descriptor

"iid" - event handler call info

OUTPUTS: .

4.5.4.1.2.2 Local data elements.

"msgLen" - length of the PDU received

"i" - counter

"readBuf" - buffer read from the pipe sourceFD

"pduhdr" - PDU header taken from readBuf

"string" - temp string to hold the formatted message info string

"messageString" - temp string to hold message to be displayed upon arrival of a message

"index" - index for messages

4.5.4.1.2.3 Interrupts and signals.

4.5.4.1.2.4 Algorithms.

4.5.4.1.2.5 Error handling.

4.5.4.1.2.6 Data conversion.

4.5.4.1.2.7 Use of other elements. Other elements that are used by the getIncomingMsg CSU:

"numMessages" - number of messages currently in the message queue

"smdForm" - widget id of the currently displayed entry form

"messages" - global queue containing messages

"numUnreadMsgs" - number of unread messages in the queue

"msgCount" - count of total messages

"msgListInfo" - array containing the mail box headers for all the messages

4.5.4.1.2.8 Logic flow. Single pass.

4.5.4.1.2.9 Data structures.

4.5.4.1.2.10 Local data files or database.

4.5.4.1.2.11 Limitations.

4.5.5 Sub-Level CSC groupListCB.

The following paragraphs under this section describe the relationship of the groupListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

#### **4.5.5.1 CSU GrpListAddCB requirements.**

The GrpListAddCB CSU contains the callback for the group list entry screen add button. It takes the entry from the text field and adds it to the group list.

##### **4.5.5.1.1 CSU GrpListAddCB design specification/constraints.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.5.1.2 CSU GrpListAddCB design.**

The information identified below represents the detailed design of the GrpListAddCB CSU.

###### **4.5.5.1.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

###### **4.5.5.1.2.2 Local data elements.**

"text" - contains the text of the highlighted item

"numItems" - contains the number of items in the list

###### **4.5.5.1.2.3 Interrupts and signals. None**

###### **4.5.5.1.2.4 Algorithms. None**

###### **4.5.5.1.2.5 Error handling. None**

###### **4.5.5.1.2.6 Data conversion. None**

###### **4.5.5.1.2.7 Use of other elements. Other elements that are used by the GrpListAddCB CSU:**

"grpListList" - widget id of the grpress list

"grpListNewItem" - widget id of the new grpress text entry field

"grpListAddBtn" - widget id for the add button on the grpress list screen

"grpListDeleteBtn" - widget id for the delete button on the grpress list screen

"grpressList" - array with all of the entries in the grpress list

- 4.5.5.1.2.8 Logic flow. Single pass.
- 4.5.5.1.2.9 Data structures. None
- 4.5.5.1.2.10 Local data files or database. None
- 4.5.5.1.2.11 Limitations. None

#### 4.5.5.2 CSU GrpListDeleteCB

The GrpListDeleteCB CSU contains the callback for the group list entry screen delete button. It takes the entry that is highlighted and deletes it from the group list.

##### 4.5.5.2.1 CSU GrpListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.5.2.2 CSU GrpListDeleteCB design.

The information identified below represents the detailed design of the GrpListDeleteCB CSU.

###### 4.5.5.2.2.1 Input/output data elements.

###### INPUTS:

- "w" contains widget id of button pressed
- "client" contains client data
- "call" contains X Windows data about the call reason

###### OUTPUTS:

###### 4.5.5.2.2.2 Local data elements.

- "i" - counter for loop
- "itemNum" - the item number from the list to be deleted
- "numItems" - the number of items in the list

###### 4.5.5.2.2.3 Interrupts and signals. None

###### 4.5.5.2.2.4 Algorithms. None

4.5.5.2.2.5 Error handling. None

4.5.5.2.2.6 Data conversion. None

4.5.5.2.2.7 Use of other elements. Other elements that are used by the GrpListDeleteCB CSU:

"grpListList" - widget id of the grpress list

"grpListAddBtn" - widget id for the add button on the grpress list screen

"grpListDeleteBtn" - widget id for the delete button on the grpress list screen

"grpressList" - array with all of the entries in the grpress list

4.5.5.2.2.8 Logic flow. Single pass.

4.5.5.2.2.9 Data structures. None

4.5.5.2.2.10 Local data files or database. None

4.5.5.2.2.11 Limitations. None

#### 4.5.5.3 CSU GrpListSaveExitCB

The GrpListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Group List screens and remanages the SysMain screens.

##### 4.5.5.3.1 CSU GrpListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.5.3.2 CSU GrpListSaveExitCB design.

The information identified below represents the detailed design of the GrpListSaveExitCB CSU.

###### 4.5.5.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:**

- 4.5.5.3.2.2 Local data elements. None
- 4.5.5.3.2.3 Interrupts and signals. None
- 4.5.5.3.2.4 Algorithms. None
- 4.5.5.3.2.5 Error handling. None
- 4.5.5.3.2.6 Data conversion. None
- 4.5.5.3.2.7 Use of other elements. Other elements that are used by the GrpListSaveExitCB CSU:
- 4.5.5.3.2.8 Logic flow. Single pass.
- 4.5.5.3.2.9 Data structures. None
- 4.5.5.3.2.10 Local data files or database. None
- 4.5.5.3.2.11 Limitations. None

**4.5.5.4 CSU GrpListClearExitCB**

The GrpListClearExitCB CSU handles the callback from the Clear and Exit pushbutton. It unmanages the Group List screens and remanages the SysMain screens.

**4.5.5.4.1 CSU GrpListClearExitCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.5.4.2 CSU GrpListClearExitCB design.**

The information identified below represents the detailed design of the GrpListClearExitCB CSU.

**4.5.5.4.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed



"client" contains client data

"call" contains X Windows data about the call reason

#### OUTPUTS:

4.5.5.4.2.2 Local data elements. None

4.5.5.4.2.3 Interrupts and signals. None

4.5.5.4.2.4 Algorithms. None

4.5.5.4.2.5 Error handling. None

4.5.5.4.2.6 Data conversion. None

4.5.5.4.2.7 Use of other elements. Other elements that are used by the GrpListClearExitCB CSU:

4.5.5.4.2.8 Logic flow. Single pass.

4.5.5.4.2.9 Data structures. None

4.5.5.4.2.10 Local data files or database. None

4.5.5.4.2.11 Limitations. None

#### 4.5.5.5 CSU PutupGroupList

The PutupGroupList CSU creates the Group List entry screen if necessary and manages the objects with the first item of the list highlighted.

##### 4.5.5.5.1 CSU PutupGroupList design requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.5.5.2 CSU PutupGroupList design.

The information identified below represents the detailed design of the main CSU.

**4.5.5.5.2.1    Input/output data elements.****INPUTS:****OUTPUTS:****4.5.5.5.2.2    Local data elements. None****4.5.5.5.2.3    Interrupts and signals. None****4.5.5.5.2.4    Algorithms. None****4.5.5.5.2.5    Error handling. None****4.5.5.5.2.6    Data conversion.****4.5.5.5.2.7    Use of other elements. Other elements that are used by the PutupGroupList CSU:****"grpListForm" - widget id for the grpress list entry form****"grpListTmi" - widget id for the grpress list TMI form****"grpListList" - widget id of the grpress list****4.5.5.5.2.8    Logic flow. Single pass.****4.5.5.5.2.9    Data structures. None****4.5.5.5.2.10   Local data files or database. None****4.5.5.5.2.11   Limitations. None****4.5.6   Sub-Level CSC listUtilities.**

The following paragraphs under this section describe the relationship of the listUtilities Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.6.1        CSU ListUpCB**

The ListUpCB CSU handles the callback from the Up pushbutton. It changes the highlighted entry to be the previous item on the list or the last item on the list if the first item is currently highlighted.

**4.5.6.1.1 CSU ListUpCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.6.1.2 CSU ListUpCB design.**

The information identified below represents the detailed design of the ListUpCB CSU.

**4.5.6.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains the list widget to manipulate

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.6.1.2.2 Local data elements.**

"pos" - the position in the list widget of the highlighted item

"count" - the number of items highlighted

"list" - widget id of the list widget to be rotated

**4.5.6.1.2.3 Interrupts and signals. None****4.5.6.1.2.4 Algorithms. None****4.5.6.1.2.5 Error handling. None****4.5.6.1.2.6 Data conversion. None****4.5.6.1.2.7 Use of other elements. Other elements that are used by the ListUpCB CSU:****4.5.6.1.2.8 Logic flow. Single pass.****4.5.6.1.2.9 Data structures. None****4.5.6.1.2.10 Local data files or database. None****4.5.6.1.2.11 Limitations. None**

#### 4.5.6.2 CSU ListDownCB

The ListDownCB CSU handles the callback from the Down pushbutton. It changes the highlighted entry to be the next item on the list or the first item on the list if the last item is currently highlighted.

##### 4.5.6.2.1 CSU ListDownCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.6.2.2 CSU ListDownCB design.

The information identified below represents the detailed design of the ListDownCB CSU.

###### 4.5.6.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains the list widget to manipulate

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.6.2.2.2 Local data elements.

"pos" - the position in the list widget of the highlighted item

"count" - the number of items highlighted

"list" - widget id of the list widget to be rotated

"numItems" - the total number of items in the list

###### 4.5.6.2.2.3 Interrupts and signals. None

###### 4.5.6.2.2.4 Algorithms. None

###### 4.5.6.2.2.5 Error handling. None

###### 4.5.6.2.2.6 Data conversion. None

###### 4.5.6.2.2.7 Use of other elements. Other elements that are used by the ListDownCB CSU:

###### 4.5.6.2.2.8 Logic flow. Single pass.

4.5.6.2.2.9 Data structures. None

4.5.6.2.2.10 Local data files or database. None

4.5.6.2.2.11 Limitations. None

#### 4.5.6.3 CSU ListAddItem

The ListAddItem CSU adds a given item into a given list widget. It adds the item to the end of the list. It checks to make sure the string is not empty.

##### 4.5.6.3.1 CSU ListAddItem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.6.3.2 CSU ListAddItem design.

The information identified below represents the detailed design of the ListAddItem CSU.

###### 4.5.6.3.2.1 Input/output data elements.

###### INPUTS:

- "list" - widget id of the list the item is to be added to
- "textItem" - widget id of the text field the string to be added is to be taken from
- "addWidget" - widget id of the add entry pushbutton
- "delWidget" - widget id of the delete entry pushbutton
- "maxItems" - maximum number of items allowed in the list

###### OUTPUTS:

- "text" - the text added to the list

###### 4.5.6.3.2.2 Local data elements.

- "text" - text to be added to the list
- "tempText" - temporary string holding the text to be added to the list
- "nullChar" - empty string
- "numItems" - the total number of items in the list

4.5.6.3.2.3 Interrupts and signals. None

4.5.6.3.2.4 Algorithms. None

4.5.6.3.2.5 Error handling. The routine makes sure that the string is not empty.

4.5.6.3.2.6 Data conversion. None

4.5.6.3.2.7 Use of other elements. Other elements that are used by the ListAddItem CSU:

4.5.6.3.2.8 Logic flow. Single pass.

4.5.6.3.2.9 Data structures. None

4.5.6.3.2.10 Local data files or database. None

4.5.6.3.2.11 Limitations. None

#### 4.5.6.4 CSU ListDeleteItem

The ListDeleteItem CSU deletes the highlighted item from a list widget.

##### 4.5.6.4.1 CSU ListDeleteItem requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.6.4.2 CSU ListDeleteItem design.

The information identified below represents the detailed design of the ListDeleteItem CSU.

###### 4.5.6.4.2.1 Input/output data elements.

###### INPUTS:

"list" - widget id of the list the item is to be added to

"addWidget" - widget id of the add entry pushbutton

"delWidget" - widget id of the delete entry pushbutton

###### OUTPUTS:

"deletedPos" - position of item deleted from the list

**4.5.6.4.2.2 Local data elements.**

"pos" - position of item highlighted

"count" - number of items highlighted

"deletedPos" - position of item deleted

"numItems" - the total number of items in the list

**4.5.6.4.2.3 Interrupts and signals. None****4.5.6.4.2.4 Algorithms. None****4.5.6.4.2.5 Error handling. None****4.5.6.4.2.6 Data conversion. None****4.5.6.4.2.7 Use of other elements. Other elements that are used by the ListDeleteItem CSU:****4.5.6.4.2.8 Logic flow. Single pass.****4.5.6.4.2.9 Data structures. None****4.5.6.4.2.10 Local data files or database. None****4.5.6.4.2.11 Limitations. None****4.5.7 Sub-Level CSC locListCB.**

The following paragraphs under this section describe the relationship of the locListCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.7.1 CSU LocListAddCb**

The LocListAddCb CSU contains the callback for the location list entry screen add button. It takes the entry from the text field and adds it to the location list.

**4.5.7.1.1 CSU LocListAddCb requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.7.1.2 CSU LocListAddCB design.**

The information identified below represents the detailed design of the LocListAddCB CSU.

**4.5.7.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.7.1.2.2 Local data elements.**

"text" - contains the text of the highlighted item

"numItems" - contains the number of items in the list

**4.5.7.1.2.3 Interrupts and signals. None****4.5.7.1.2.4 Algorithms. None****4.5.7.1.2.5 Error handling. None****4.5.7.1.2.6 Data conversion. None****4.5.7.1.2.7 Use of other elements. Other elements that are used by the LocListAddCB CSU:**

"locListList" - widget id of the locess list

"locListNewItem" - widget id of the new locess text entry field

"locListAddBtn" - widget id for the add button on the locess list screen

"locListDeleteBtn" - widget id for the delete button on the locess list screen

"locessList" - array with all of the entries in the locess list

**4.5.7.1.2.8 Logic flow. Single pass.****4.5.7.1.2.9 Data structures. None****4.5.7.1.2.10 Local data files or database. None****4.5.7.1.2.11 Limitations. None**



#### 4.5.7.2 CSU LocListDeleteCB

The LocListDeleteCB CSU contains the callback for the location list entry screen delete button. It takes the entry that is highlighted and deletes it from the location list.

##### 4.5.7.2.1 CSU LocListDeleteCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.7.2.2 CSU LocListDeleteCB design.

The information identified below represents the detailed design of the LocListDeleteCB CSU.

###### 4.5.7.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS:

###### 4.5.7.2.2.2 Local data elements.

"i" - counter for loop

"itemNum" - the item number from the list to be deleted

"numItems" - the number of items in the list

###### 4.5.7.2.2.3 Interrupts and signals. None

###### 4.5.7.2.2.4 Algorithms. None

###### 4.5.7.2.2.5 Error handling. None

###### 4.5.7.2.2.6 Data conversion. None

###### 4.5.7.2.2.7 Use of other elements. Other elements that are used by the LocListDeleteCB CSU:

"locListList" - widget id of the locess list

"locListAddBtn" - widget id for the add button on the locess list screen

"locListDeleteBtn" - widget id for the delete button on the locess list screen

"locessList" - array with all of the entries in the locess list

4.5.7.2.2.8 Logic flow. Single pass.

4.5.7.2.2.9 Data structures. None

4.5.7.2.2.10 Local data files or database. None

4.5.7.2.2.11 Limitations. None

#### 4.5.7.3 CSU LocListSaveExitCB

The LocListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Location List screens and remanages the SysMain screens.

##### 4.5.7.3.1 CSU LocListSaveExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.7.3.2 CSU LocListSaveExitCB design.

The information identified below represents the detailed design of the LocListSaveExitCB CSU.

###### 4.5.7.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS:

4.5.7.3.2.2 Local data elements. None

4.5.7.3.2.3 Interrupts and signals. None

4.5.7.3.2.4 Algorithms. None

4.5.7.3.2.5 Error handling. None

4.5.7.3.2.6 Data conversion. None

4.5.7.3.2.7 Use of other elements. Other elements that are used by the LocListSaveExitCB CSU:

4.5.7.3.2.8 Logic flow. Single pass.

4.5.7.3.2.9 Data structures. None

4.5.7.3.2.10 Local data files or database. None

4.5.7.3.2.11 Limitations. None

#### 4.5.7.4 CSU LocListClearExitCB .

The LocListSaveExitCB CSU handles the callback from the Save and Exit pushbutton. It unmanages the Location List screens and remanages the SysMain screens.

##### 4.5.7.4.1 CSU LocListClearExitCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.7.4.2 CSU LocListClearExitCB design.

The information identified below represents the detailed design of the LocListClearExitCB CSU.

###### 4.5.7.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.7.4.2.2 Local data elements. None

4.5.7.4.2.3 Interrupts and signals. None

4.5.7.4.2.4 Algorithms. None

- 4.5.7.4.2.5 Error handling. None
- 4.5.7.4.2.6 Data conversion. None
- 4.5.7.4.2.7 Use of other elements. Other elements that are used by the LocListClearExitCB CSU:
- 4.5.7.4.2.8 Logic flow. Single pass.
- 4.5.7.4.2.9 Data structures. None
- 4.5.7.4.2.10 Local data files or database. None
- 4.5.7.4.2.11 Limitations. None

#### 4.5.7.5 CSU PutupLocationList .

The PutupLocationList CSU creates the Location List entry screen if necessary and manages the objects with the first item of the list highlighted.

##### 4.5.7.5.1 CSU PutupLocationList requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.7.5.2 CSU PutupLocationList design.

The information identified below represents the detailed design of the main CSU.

###### 4.5.7.5.2.1 Input/output data elements.

INPUTS:

OUTPUTS:

- 4.5.7.5.2.2 Locational data elements. None
- 4.5.7.5.2.3 Interrupts and signals. None
- 4.5.7.5.2.4 Algorithms. None

4.5.7.5.2.5 Error handling. None

4.5.7.5.2.6 Data conversion.

4.5.7.5.2.7 Use of other elements. Other elements that are used by the PutupLocationList CSU:

"locListForm" - widget id for the locess list entry form

"locListTmi" - widget id for the locess list TMI form

"locListList" - widget id of the locess list

4.5.7.5.2.8 Logic flow. Single pass.

4.5.7.5.2.9 Data structures. None

4.5.7.5.2.10 Locational data files or database. None

4.5.7.5.2.11 Limitations. None

#### **4.5.8 Sub-Level CSC logonCB.**

The following paragraphs under this section describe the relationship of the logonCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.8.1 CSU LogonNetLogonCB**

The LogonNetLogonCB CSU handles the callback for the logon pushbutton. It calls dmslogin to inform dms that it exists and sets up the process for receiving messages via a pipe.

##### **4.5.8.1.1 CSU LogonNetLogonCB requirements**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.8.1.2 CSU LogonNetLogonCB design.**

The information identified below represents the detailed design of the LogonNetLogonCB CSU.

**4.5.8.1.2.1    Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed  
"client" contains client data  
"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.8.1.2.2    Local data elements.**

"exerID" - string with the exercise id  
"logonName" - string with logon name  
"exerIDtemp" - temp with integer version of the exercise id  
"offset\_file" - file descriptor for the file containing the zulu time offset  
"nullChar" - empty string

**4.5.8.1.2.3    Interrupts and signals. None****4.5.8.1.2.4    Algorithms. None****4.5.8.1.2.5    Error handling. None****4.5.8.1.2.6    Data conversion. None****4.5.8.1.2.7    Use of other elements. Other elements that are used by the LogonNetLogonCB CSU:**

"exerIDnum" - global with the exercise ID number  
"logonNameText" - widget id of the text field containing the logon name  
"exerciseIDText" - widget id of the text field containing the exercise id  
"groupList" - global containing the CEOI/group list  
"OFFSET\_FILENAME" - define with the name of the offset file  
"pipeFD" - file descriptor of the pipe over which messages will come in  
"sysMainButton" - widget id of sys main pushbutton  
"logonTmi" - widget id of the TMI area of the logon screen  
"logonForm" - widget id of the entry area of the logon screen

**4.5.8.1.2.8    Logic flow. Single pass.****4.5.8.1.2.9    Data structures. None****4.5.8.1.2.10   Local data files or database. None****4.5.8.1.2.11   Limitations. None**

#### 4.5.8.2 CSU PutupLogon

The PutupLogon CSU creates the logon screen if necessary and manages the objects.

##### 4.5.8.2.1 CSU PutupLogon requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.8.2.2 CSU PutupLogon design.

The information identified below represents the detailed design of the PutupLogon CSU.

###### 4.5.8.2.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.8.2.2.2 Local data elements. None

###### 4.5.8.2.2.3 Interrupts and signals. None

###### 4.5.8.2.2.4 Algorithms. None

###### 4.5.8.2.2.5 Error handling. None

###### 4.5.8.2.2.6 Data conversion. None

###### 4.5.8.2.2.7 Use of other elements. Other elements that are used by the PutupLogon CSU:

"logonTmi" - widget id of the TMI area of the logon screen

"logonForm" - widget id of the entry area of the logon screen

###### 4.5.8.2.2.8 Logic flow. Single pass.

###### 4.5.8.2.2.9 Data structures. None

###### 4.5.8.2.2.10 Local data files or database. None

###### 4.5.8.2.2.11 Limitations. None

#### 4.5.9 Sub-Level CSC main.

The following paragraphs under this section describe the relationship of the main Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.9.1 CSU main

The main CSU initializes the X window client, all variables to be used by the report menus, creates the toplevel form and goes into a loop waiting for stuff to happen.

##### 4.5.9.1.1 CSU main requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.9.1.2 CSU main design.

The information identified below represents the detailed design of the main CSU.

##### 4.5.9.1.2.1 Input/output data elements.

###### INPUTS:

"argc" - number of arguments

"argv" - array of argument strings

OUTPUTS: None

##### 4.5.9.1.2.2 Local data elements.

"display" - the X display being used

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

##### 4.5.9.1.2.3 Interrupts and signals. None

##### 4.5.9.1.2.4 Algorithms. None

##### 4.5.9.1.2.5 Error handling. None

##### 4.5.9.1.2.6 Data conversion. None



4.5.9.1.2.7 Use of other elements. Other elements that are used by the main CSU:

"context" - X windows context of the application

"numMessages" - number of messages in the message queue

"numUnreadMsgs" - number of unread messages in the message queue

"msgCount" - how high the messages can count

"AppShell" - widget id of the application shell

"Shell000" - widget id of the top level popup shell

"Form" - widget id of the top level form

4.5.9.1.2.8 Logic flow. Single pass with infinite loop at the end.

4.5.9.1.2.9 Data structures. None

4.5.9.1.2.10 Local data files or database. None

4.5.9.1.2.11 Limitations. None

#### 4.5.10 Sub-Level CSC movcmdTmi\_call

The following paragraphs under this section describe the relationship of the movcmdTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.10.1 CSU MovcmdClearNRetCB

The MovcmdClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Movcmd window and remanage the Reports window.

##### 4.5.10.1.1 CSU MovcmdClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.10.1.2 CSU MovcmdClearNRetCB design.

The information identified below represents the detailed design of the MovcmdClearNRetCB CSU.

**4.5.10.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.10.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.10.1.2.3 Interrupts and signals. None****4.5.10.1.2.4 Algorithms. None****4.5.10.1.2.5 Error handling. None****4.5.10.1.2.6 Data conversion. None****4.5.10.1.2.7 Use of other elements. Other elements that are used by the MovcmdClearNRetCB CSU:**

"MovcmdTmiForm" - form widget containing Movcmd TMI

"MovcmdForm" - form widget containing Movcmd Entry area

"cikLabel2" - label widget

**4.5.10.1.2.8 Logic flow. Single pass****4.5.10.1.2.9 Data structures. None****4.5.10.1.2.10 Local data files or database. None****4.5.10.1.2.11 Limitations. None****4.5.10.2 CSU MovcmdSaveNRetCB .**

The MovcmdSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Movcmd window and remanage the Reports window.

**4.5.10.2.1 CSU MovcmdSaveNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.10.2.2 CSU MovcmdSaveNRetCB design.**

The information identified below represents the detailed design of the MovcmdSaveNRetCB CSU.

**4.5.10.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.10.2.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

**4.5.10.2.2.3 Interrupts and signals. None****4.5.10.2.2.4 Algorithms. None****4.5.10.2.2.5 Error handling. None****4.5.10.2.2.6 Data conversion. None****4.5.10.2.2.7 Use of other elements. Other elements that are used by the MovcmdSaveNRetCB CSU:**

"MovcmdTmiForm" - form widget containing Movcmd TMI

"MovcmdForm" - form widget containing Movcmd Entry area

"cikLabel2" - label widget

"cikLabel1" - label widget

**4.5.10.2.2.8 Logic flow. Single pass**

4.5.10.2.2.9 Data structures. None

4.5.10.2.2.10 Local data files or database. None

4.5.10.2.2.11 Limitations. None

#### 4.5.10.3 CSU MovcmdSndRoutCB

The MovcmdSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls MovcmdSend with priority ROUTINE.

##### 4.5.10.3.1 CSU MovcmdSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.10.3.2 CSU MovcmdSndRoutCB design.

The information identified below represents the detailed design of the MovcmdSndRoutCB CSU.

###### 4.5.10.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.10.3.2.2 Local data elements. None

4.5.10.3.2.3 Interrupts and signals. None

4.5.10.3.2.4 Algorithms. None

4.5.10.3.2.5 Error handling. None

4.5.10.3.2.6 Data conversion. None

4.5.10.3.2.7 Use of other elements. Other elements that are used by the MovcmdSndRoutCB CSU:

**"PRIORITY\_ROUTINE" - define for priority message**

4.5.10.3.2.8 Logic flow. Single pass

4.5.10.3.2.9 Data structures. None

4.5.10.3.2.10 Local data files or database. None

4.5.10.3.2.11 Limitations. None

**4.5.10.4 CSU MovcmdSend**

The MovcmdSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldMovcmd to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

**4.5.10.4.1 CSU MovcmdSend requirements.**

The MovcmdSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

**4.5.10.4.2 CSU MovcmdSend design.**

The information identified below represents the detailed design of the MovcmdSend CSU.

**4.5.10.4.2.1 Input/output data elements.****INPUTS:**

"prio" - priority of the message being sent

**OUTPUTS: None****4.5.10.4.2.2 Local data elements.**

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

"ObserverLocation" - location of observer

"TargetID" - ID of the target

"MoTask" - movement task

"Who" - who to move

"MoWhen" - when to move

"dateString" - string with date info

"DTG" - structure with the date info

4.5.10.4.2.3 Interrupts and signals. None

4.5.10.4.2.4 Algorithms. None

4.5.10.4.2.5 Error handling. None

4.5.10.4.2.6 Data conversion. None

4.5.10.4.2.7 Use of other elements. Other elements that are used by the MovcmdSend CSU:

"MovcmdAdd" - information on the Movcmd address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call

"When" - information on the When selection menu

"Task" - information on the Task selection menu

"reuseFlag" - boolean; is this a reuse and include

4.5.10.4.2.8 Logic flow. Single pass

4.5.10.4.2.9 Data structures. None

4.5.10.4.2.10 Local data files or database. None

4.5.10.4.2.11 Limitations. None

#### 4.5.10.5 CSU MovcmdSndUrgCB design specification/constraints.

The MovcmdSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls MovcmdSend with priority URGENT.

##### 4.5.10.5.1 CSU MovcmdSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.10.5.2 CSU MovcmdSndUrgCB design.

The information identified below represents the detailed design of the MovcmdSndUrgCB CSU.

**4.5.10.5.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.10.5.2.2 Local data elements. None****4.5.10.5.2.3 Interrupts and signals. None****4.5.10.5.2.4 Algorithms. None****4.5.10.5.2.5 Error handling. None****4.5.10.5.2.6 Data conversion. None****4.5.10.5.2.7 Use of other elements. Other elements that are used by the MovcmdSndUrgCB CSU:  
"PRIORITY\_URGENT" - define for urgent priority****4.5.10.5.2.8 Logic flow. Single pass****4.5.10.5.2.9 Data structures. None****4.5.10.5.2.10 Local data files or database. None****4.5.10.5.2.11 Limitations. None****4.5.10.6 CSU ClearMovcmd**

The ClearMovcmd CSU clears all the entries of the Movcmd report generation screen.

**4.5.10.6.1 CSU ClearMovcmd requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.10.6.2 CSU ClearMovcmd design.**

The information identified below represents the detailed design of the ClearMovcmd CSU.

**4.5.10.6.2.1 Input/output data elements.**

INPUTS:

OUTPUTS: None

**4.5.10.6.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

**4.5.10.6.2.3 Interrupts and signals. None****4.5.10.6.2.4 Algorithms. None****4.5.10.6.2.5 Error handling. None****4.5.10.6.2.6 Data conversion. None****4.5.10.6.2.7 Use of other elements. Other elements that are used by the ClearMovcmd CSU:**

"MovcmdAdd" - info on the selection menu for the address

"EnemyType" - info on the selection menu for the enemy type

"EnemyActivity" - info on the selection menu for the enemy activity

"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions

"SpeedText" - widget id for speed text

"NumText" - widget id for number text

"UnitText" - widget id for unit text

"cikText1"- widget id for cik1 text

"cikText2"- widget id for cik2 text

**4.5.10.6.2.8 Logic flow. Single pass****4.5.10.6.2.9 Data structures. None**



4.5.10.6.2.10 Local data files or database. None

4.5.10.6.2.11 Limitations. None

#### **4.5.11 Sub-Level CSC movcmd\_call.**

The following paragraphs under this section describe the relationship of the movcmd\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.11.1 CSU lctn\_moreCB**

The lctn\_moreCB CSU handles the callback for the lctn More pushbutton. It displays additional selections if they exist.

##### **4.5.11.1.1 CSU lctn\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.11.1.2 CSU lctn\_moreCB design.**

The information identified below represents the detailed design of the lctn\_moreCB CSU.

##### **4.5.11.1.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

##### **4.5.11.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

##### **4.5.11.1.2.3 Interrupts and signals. None**

4.5.11.1.2.4 Algorithms. None

4.5.11.1.2.5 Error handling. None

4.5.11.1.2.6 Data conversion. None

4.5.11.1.2.7 Use of other elements. Other elements that are used by the lctn\_moreCB CSU:  
"Lctn" - info on Enemy Activity selection menu

4.5.11.1.2.8 Logic flow. Single pass

4.5.11.1.2.9 Data structures. None

4.5.11.1.2.10 Local data files or database. None

4.5.11.1.2.11 Limitations. None

#### 4.5.11.2 CSU lctn\_rotateCB

The lctn\_rotateCB CSU handles the callback for the lctn Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.11.2.1 CSU lctn\_rotateCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.11.2.2 CSU lctn\_rotateCB design.

The information identified below represents the detailed design of the lctn\_rotateCB CSU.

###### 4.5.11.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

- 4.5.11.2.2.2 Local data elements. None
- 4.5.11.2.2.3 Interrupts and signals. None
- 4.5.11.2.2.4 Algorithms. None
- 4.5.11.2.2.5 Error handling. None
- 4.5.11.2.2.6 Data conversion. None
- 4.5.11.2.2.7 Use of other elements. Other elements that are used by the lctn\_rotateCB CSU:  
"Lctn" - info on the enemy activity selection menu
- 4.5.11.2.2.8 Logic flow. Single pass
- 4.5.11.2.2.9 Data structures. None
- 4.5.11.2.2.10 Local data files or database. None
- 4.5.11.2.2.11 Limitations. None

#### 4.5.11.3 CSU task\_moreCB

The task\_moreCB CSU handles the callback for the task More pushbutton. It displays additional selections if they exist.

##### 4.5.11.3.1 CSU task\_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.11.3.2 CSU task\_moreCB design.

The information identified below represents the detailed design of the task\_moreCB CSU.

###### 4.5.11.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.11.3.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.11.3.2.3 Interrupts and signals. None****4.5.11.3.2.4 Algorithms. None****4.5.11.3.2.5 Error handling. None****4.5.11.3.2.6 Data conversion. None****4.5.11.3.2.7 Use of other elements. Other elements that are used by the task\_moreCB CSU:**

"Task" - info on Enemy Activity selection menu

**4.5.11.3.2.8 Logic flow. Single pass****4.5.11.3.2.9 Data structures. None****4.5.11.3.2.10 Local data files or database. None****4.5.11.3.2.11 Limitations. None****4.5.11.4 CSU task\_rotateCB**

The task\_rotateCB CSU handles the callback for the task Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.11.4.1 CSU task\_rotateCB requirements**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.5.11.4.2 CSU task\_rotateCB design.

The information identified below represents the detailed design of the task\_rotateCB CSU.

##### 4.5.11.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.11.4.2.2 Local data elements. None

4.5.11.4.2.3 Interrupts and signals. None .

4.5.11.4.2.4 Algorithms. None

4.5.11.4.2.5 Error handling. None

4.5.11.4.2.6 Data conversion. None

4.5.11.4.2.7 Use of other elements. Other elements that are used by the task\_rotateCB CSU:

"Task" - info on the enemy activity selection menu

4.5.11.4.2.8 Logic flow. Single pass

4.5.11.4.2.9 Data structures. None

4.5.11.4.2.10 Local data files or database. None

4.5.11.4.2.11 Limitations. None

#### 4.5.11.5 CSU when\_rotateCB

The when\_rotateCB CSU handles the callback for the when Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.11.5.1 CSU when\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.11.5.2 CSU when\_rotateCB design.**

The information identified below represents the detailed design of the when\_rotateCB CSU.

**4.5.11.5.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

4.5.11.5.2.2 Local data elements. None

4.5.11.5.2.3 Interrupts and signals. None

4.5.11.5.2.4 Algorithms. None

4.5.11.5.2.5 Error handling. None

4.5.11.5.2.6 Data conversion. None

4.5.11.5.2.7 Use of other elements. Other elements that are used by the when\_rotateCB CSU:

"When" - info on the enemy activity selection menu

4.5.11.5.2.8 Logic flow. Single pass

4.5.11.5.2.9 Data structures. None

4.5.11.5.2.10 Local data files or database. None

4.5.11.5.2.11 Limitations. None

#### **4.5.11.6 CSU movcmdad\_moreCB**

The movcmdad\_moreCB CSU handles the callback for the movcmdad More pushbutton. It displays additional selections if they exist.

##### **4.5.11.6.1 CSU movcmdad\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.11.6.2 CSU movcmdad\_moreCB design.**

The information identified below represents the detailed design of the movcmdad\_moreCB CSU.

###### **4.5.11.6.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: .**

###### **4.5.11.6.2.2 Local data elements. None**

###### **4.5.11.6.2.3 Interrupts and signals. None**

###### **4.5.11.6.2.4 Algorithms. None**

###### **4.5.11.6.2.5 Error handling. None**

###### **4.5.11.6.2.6 Data conversion. None**

###### **4.5.11.6.2.7 Use of other elements. Other elements that are used by the movcmdad\_moreCB CSU:**

"MovcmdAdd" - info on the Movcmd address selection menu

###### **4.5.11.6.2.8 Logic flow. Single pass**

###### **4.5.11.6.2.9 Data structures. None**

###### **4.5.11.6.2.10 Local data files or database. None**

**4.5.11.6.2.11 Limitations. None****4.5.11.7 CSU movcmdad\_rotateCB**

The movcmdad\_rotateCB CSU handles the callback for the movcmdad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.11.7.1 CSU movcmdad\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.11.7.2 CSU movcmdad\_rotateCB design.**

The information identified below represents the detailed design of the movcmdad\_rotateCB CSU.

**4.5.11.7.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .****4.5.11.7.2.2 Local data elements. None****4.5.11.7.2.3 Interrupts and signals. None****4.5.11.7.2.4 Algorithms. None****4.5.11.7.2.5 Error handling. None****4.5.11.7.2.6 Data conversion. None****4.5.11.7.2.7 Use of other elements. Other elements that are used by the movcmdad\_rotateCB CSU:**

"MovcmdAdd" - info on Movcmd address selection menu

**4.5.11.7.2.8 Logic flow. Single pass**



4.5.11.7.2.9 Data structures. None

4.5.11.7.2.10 Local data files or database. None

4.5.11.7.2.11 Limitations. None

#### 4.5.11.8 CSU PutupMovcmd

The PutupMovcmd CSU will create the Movcmd report screen if necessary and will also manage the screen.

##### 4.5.11.8.1 CSU PutupMovcmd requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.11.8.2 CSU PutupMovcmd design.

The information identified below represents the detailed design of the PutupMovcmd CSU.

###### 4.5.11.8.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.11.8.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

###### 4.5.11.8.2.3 Interrupts and signals. None

###### 4.5.11.8.2.4 Algorithms. None

###### 4.5.11.8.2.5 Error handling. None

###### 4.5.11.8.2.6 Data conversion. None

**4.5.11.8.2.7 Use of other elements.** Other elements that are used by the PutupMovcmd CSU:

"MovcmdForm" - widget id of movcmd form

"MovcmdAdd" - info on Movcmd address selection menu

"addressList" - list of addresses

"MovcmdMoreLabel" - widget id of More label

"MovcmdMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply

"MovcmdTmiForm" widget id of movcmd TMI form

**4.5.11.8.2.8 Logic flow.** Single pass

**4.5.11.8.2.9 Data structures.** None

**4.5.11.8.2.10 Local data files or database.** None

**4.5.11.8.2.11 Limitations.** None

#### **4.5.12 Sub-Level CSC msg1CB.**

The following paragraphs under this section describe the relationship of the msg1CB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.12.1 CSU MsgsPrevCB**

The MsgsPrevCB CSU handles the callback from the previous pushbutton. The previous message in the list is highlighted.

##### **4.5.12.1.1 CSU MsgsPrevCB requirements.**

The MsgsPrevCB CSU satisfies section 3.2.1.2.2.3.1.1 of the specific requirements of the DMCC system.

##### **4.5.12.1.2 CSU MsgsPrevCB design.**

The information identified below represents the detailed design of the MsgsPrevCB CSU.

##### **4.5.12.1.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

**4.5.12.1.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

**4.5.12.1.2.3 Interrupts and signals. None**

**4.5.12.1.2.4 Algorithms. None**

**4.5.12.1.2.5 Error handling. None**

**4.5.12.1.2.6 Data conversion. None**

**4.5.12.1.2.7 Use of other elements. Other elements that are used by the MsgsPrevCB CSU:**

"msg1List" - widget id of the msg1 list

**4.5.12.1.2.8 Logic flow. Single pass.**

**4.5.12.1.2.9 Data structures. None**

**4.5.12.1.2.10 Local data files or database. None**

**4.5.12.1.2.11 Limitations. None**

**4.5.12.2 CSU MsgsNextCB**

The MsgsNextCB CSU handles the callback from the next pushbutton. The next message in the list is highlighted.

**4.5.12.2.1 CSU MsgsNextCB requirements**

The MsgsNextCB CSU satisfies section 3.2.1.2.2.3.1.1 of the specific requirements of the DMCC system.

**4.5.12.2.2 CSU MsgsNextCB design.**

The information identified below represents the detailed design of the MsgsNextCB CSU.

**4.5.12.2.2.1 Input/output data elements.**

INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.12.2.2.2 Local data elements.

"pos" - position in list highlighted

"count" - number of items highlighted

"numItems" - number of items in the list

4.5.12.2.2.3 Interrupts and signals. None

4.5.12.2.2.4 Algorithms. None

4.5.12.2.2.5 Error handling. None

4.5.12.2.2.6 Data conversion. None

4.5.12.2.2.7 Use of other elements. Other elements that are used by the MsgsNextCB CSU:

"msg1List" - widget id of the msg1 list

4.5.12.2.2.8 Logic flow. Single pass.

4.5.12.2.2.9 Data structures. None

4.5.12.2.2.10 Local data files or database. None

4.5.12.2.2.11 Limitations. None

**4.5.12.3 CSU MsgsReadCB**

The MsgsReadCB CSU handles the callback for the Read pushbutton. It brings up a formatted version of the currently highlighted pdu.

**4.5.12.3.1 CSU MsgsReadCB requirements**

The MsgsReadCB CSU satisfies section 3.2.1.2.2.2.3 of the specific requirements of the DMCC system.

**4.5.12.3.2 CSU MsgsReadCB design.**

The information identified below represents the detailed design of the MsgsReadCB CSU.

**4.5.12.3.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None**

**4.5.12.3.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

"msgBuf" - message buffer

**4.5.12.3.2.3 Interrupts and signals. None****4.5.12.3.2.4 Algorithms. None****4.5.12.3.2.5 Error handling. None****4.5.12.3.2.6 Data conversion. None****4.5.12.3.2.7 Use of other elements. Other elements that are used by the MsgsReadCB CSU:**

"msg1List" - widget id of the msg1 list

"msg1Tmi" - widget id of the TMI of msg1

"msg1Form" - widget id of the entry aread of msg1

"messages" - queue with messages

"msgListIndex" - array of mappings of msg1 list entries to messages entries

"numUnreadMsgs" - number of unread messages in the queue

"msgReadText" - widget id of read screen text

**4.5.12.3.2.8 Logic flow. Single pass.****4.5.12.3.2.9 Data structures. None****4.5.12.3.2.10 Local data files or database. None****4.5.12.3.2.11 Limitations. None****4.5.12.4 CSU MsgsDeleteCB**

The MsgsDeleteCB CSU handles the callback for the Delete pushbutton. It deletes the currently highlighted pdu.

**4.5.12.4.1 CSU MsgsDeleteCB requirements.**

The MsgsDeleteCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.5.1 of the specific requirements of the DMCC system.

**4.5.12.4.2 CSU MsgsDeleteCB design.**

The information identified below represents the detailed design of the MsgsDeleteCB CSU.

**4.5.12.4.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.12.4.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

"numItems" - number of items in the list

"i" - counter

**4.5.12.4.2.3 Interrupts and signals. None****4.5.12.4.2.4 Algorithms. None****4.5.12.4.2.5 Error handling. None****4.5.12.4.2.6 Data conversion. None****4.5.12.4.2.7 Use of other elements. Other elements that are used by the MsgsDeleteCB CSU:**

"msg1List" - widget id of the msg1 list

"numUnreadMsgs" - number of unread messages in the queue

"messages" - global queue containing messages

"msgListIndex" - array of mappings of msg1 list entries to messages entries

"numMessages" - number of messages in the queue

**4.5.12.4.2.8 Logic flow. Single pass.****4.5.12.4.2.9 Data structures. None**

4.5.12.4.2.10 Local data files or database. None

4.5.12.4.2.11 Limitations. None

#### 4.5.12.5 CSU FormatMessageInfo

The FormatMessageInfo CSU takes information about the message and creates the header string for the Msg1 window.

##### 4.5.12.5.1 CSU FormatMessageInfo requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.12.5.2 CSU FormatMessageInfo design.

The information identified below represents the detailed design of the FormatMessageInfo CSU.

###### 4.5.12.5.2.1 Input/output data elements.

###### INPUTS:

"type" - type of message  
"sender" - sender of message  
"timeHour" - hour message sent  
"timeMin" - minute message sent  
"pri" - priority of message

###### OUTPUTS:

"info" - formatted string of message information

###### 4.5.12.5.2.2 Local data elements.

"info" - formatted string of message information  
"priChar" - character containing the priority of the message

###### 4.5.12.5.2.3 Interrupts and signals. None

###### 4.5.12.5.2.4 Algorithms. None

###### 4.5.12.5.2.5 Error handling. None

###### 4.5.12.5.2.6 Data conversion. None

4.5.12.5.2.7 Use of other elements. Other elements that are used by the FormatMessageInfo CSU:

None

4.5.12.5.2.8 Logic flow. Single pass.

4.5.12.5.2.9 Data structures. None

4.5.12.5.2.10 Local data files or database. None

4.5.12.5.2.11 Limitations. None

#### 4.5.12.6 CSU PutupMsg1

The PutupMsg1 CSU will create the Msg1 screen if necessary and will also manage the screen.

##### 4.5.12.6.1 CSU MsgsPrevCB requirements

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.12.6.2 CSU MsgsPrevCB design.

The information identified below represents the detailed design of the MsgsPrevCB CSU.

###### 4.5.12.6.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

4.5.12.6.2.2 Local data elements. None

4.5.12.6.2.3 Interrupts and signals. None

4.5.12.6.2.4 Algorithms. None

4.5.12.6.2.5 Error handling. None

4.5.12.6.2.6 Data conversion. None



4.5.12.6.2.7 Use of other elements. Other elements that are used by the MsgsPrevCB CSU:

"msg1List" - widget id of the msg1 list

"msg1Tmi" - widget id of the TMI of msg1

"msg1Form" - widget id of the entry aread of msg1

"smdForm" - widget id of the currently displayed entry form

4.5.12.6.2.8 Logic flow. Single pass.

4.5.12.6.2.9 Data structures. None

4.5.12.6.2.10 Local data files or database None

4.5.12.6.2.11 Limitations. None

#### 4.5.12.7 CSU TestPriority

The TestPriority CSU will take two messages and determine which one has higher priority. Priority is determined by not only the priority field, but also the timestamp of the message.

##### 4.5.12.7.1 CSU TestPriority requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.12.7.2 CSU TestPriority design.

The information identified below represents the detailed design of the TestPriority CSU.

###### 4.5.12.7.2.1 Input/output data elements.

###### INPUTS:

"index1" - first index to be tested

"index2" - second index to be tested

###### OUTPUTS:

"result" - result of the test

###### 4.5.12.7.2.2 Local data elements.

"prio1" - priority of index 1

"prio2" - priority of index 2  
"ind1" - index of index 1  
"ind2" - index of index 2  
"result" - result of the test

4.5.12.7.2.3 Interrupts and signals. None

4.5.12.7.2.4 Algorithms. None

4.5.12.7.2.5 Error handling. None

4.5.12.7.2.6 Data conversion. None

4.5.12.7.2.7 Use of other elements. Other elements that are used by the TestPriority CSU:  
"messages" - global queue containing messages

4.5.12.7.2.8 Logic flow. Single pass.

4.5.12.7.2.9 Data structures. None

4.5.12.7.2.10 Local data files or database. None

4.5.12.7.2.11 Limitations. None

#### 4.5.12.8 CSU UpdateList

The UpdateList CSU will update the Msg1 list with any new messages and order the messages appropriately.

##### 4.5.12.8.1 CSU UpdateList requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.12.8.2 CSU UpdateList design.

The information identified below represents the detailed design of the UpdateList CSU.

###### 4.5.12.8.2.1 Input/output data elements.

INPUTS: None

**OUTPUTS: None****4.5.12.8.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

"i" - counter

"listEntry" - temp string carrying the msg1 display info

"swap" - boolean for the sort

"tmp" - a temp variable

**4.5.12.8.2.3 Interrupts and signals. None****4.5.12.8.2.4 Algorithms. None****4.5.12.8.2.5 Error handling. None****4.5.12.8.2.6 Data conversion. None****4.5.12.8.2.7 Use of other elements. Other elements that are used by the UpdateList CSU:**

"numMessages" - number of messages currently in the message queue

"msgListInfo" - array containing the mail box headers for all the messages

"msgListIndex" - array of mappings of msg1 list entries to messages entries

**4.5.12.8.2.8 Logic flow. Single pass.****4.5.12.8.2.9 Data structures. None****4.5.12.8.2.10 Local data files or database. None****4.5.12.8.2.11 Limitations. None****4.5.12.9 CSU MsgsReuseCB**

The MsgsReuseCB CSU handles the callback for the Reuse pushbutton. It saves the currently highlighted PDU in a global and thing brings up the reuse report screen.

**4.5.12.9.1 CSU MsgsReuseCB requirements.**

The MsgsReuseCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

**4.5.12.9.2 CSU MsgsReuseCB design.**

The information identified below represents the detailed design of the MsgsReuseCB CSU.

**4.5.12.9.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

**4.5.12.9.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

**4.5.12.9.2.3 Interrupts and signals. None****4.5.12.9.2.4 Algorithms. None****4.5.12.9.2.5 Error handling. None****4.5.12.9.2.6 Data conversion. None****4.5.12.9.2.7 Use of other elements. Other elements that are used by the MsgsReuseCB CSU:**

"msg1List" - widget id of the msg1 list

"reusePDU" - pdu to be forwarded

"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form

"msgListIndex" - array of mappings of msg1 list entries to messages entries

**4.5.12.9.2.8 Logic flow. Single pass.****4.5.12.9.2.9 Data structures. None****4.5.12.9.2.10 Local data files or database. None**

**4.5.12.9.2.11 Limitations. None****4.5.12.10 CSU MsgsReuseNIncludeCB**

The MsgsReuseNIncludeCB CSU handles the callback for the Reuse and Include pushbutton. It saves the currently highlighted PDU in a global and thing brings up the report screen so that a new report can be generated for inclusion with the reused message.

**4.5.12.10.1 CSU MsgsReuseNIncludeCB requirements.**

The MsgsReuseNIncludeCB CSU satisfies section 3.2.1.2.2.2.3 and 3.2.1.2.2.3.2.3 of the specific requirements of the DMCC system.

**4.5.12.10.2 CSU MsgsReuseNIncludeCB design.**

The information identified below represents the detailed design of the MsgsReuseNIncludeCB CSU.

**4.5.12.10.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.12.10.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

**4.5.12.10.2.3 Interrupts and signals. None****4.5.12.10.2.4 Algorithms. None****4.5.12.10.2.5 Error handling. None****4.5.12.10.2.6 Data conversion. None****4.5.12.10.2.7 Use of other elements. Other elements that are used by the MsgsReuseNIncludeCB CSU:**

"msg1List" - widget id of the msg1 list

"reusePDU" - pdu to be forwarded

"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form

"msgListIndex" - array of mappings of msg1 list entries to messages entries

"reuseFlag" - are we including a message

4.5.12.10.2.8 Logic flow. Single pass.

4.5.12.10.2.9 Data structures. None

4.5.12.10.2.10 Local data files or database. None

4.5.12.10.2.11 Limitations. None

#### 4.5.12.11 CSU MsgsSaECB.

The MsgsSaECB CSU handles the callback from the Save and Exit pushbutton. It leaves the Msgs1 screen and returns to the SysMain screen.

##### 4.5.12.11.1 CSU MsgsSaECB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.12.11.2 CSU MsgsSaECB design.

The information identified below represents the detailed design of the MsgsSaECB CSU.

###### 4.5.12.11.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

###### 4.5.12.11.2.2 Local data elements.

"pos" - position in list highlighted

"count" - number of items highlighted

###### 4.5.12.11.2.3 Interrupts and signals. None

4.5.12.11.2.4 Algorithms. None

4.5.12.11.2.5 Error handling. None

4.5.12.11.2.6 Data conversion. None

4.5.12.11.2.7 Use of other elements. Other elements that are used by the MsgsSaECB CSU:

"msg1List" - widget id of the msg1 list

"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form

4.5.12.11.2.8 Logic flow. Single pass.

4.5.12.11.2.9 Data structures. None

4.5.12.11.2.10 Local data files or database. None

4.5.12.11.2.11 Limitations. None

#### 4.5.12.12 CSU MsgsReplyCB

The MsgsReplyCB callback handles the callback for the Reply pushbutton. It stores the CEOI of the sender of the highlighted message and then brings up the report screen for a message to be generated to the sender of said message.

##### 4.5.12.12.1 CSU MsgsReplyCB requirements.

The MsgsReplyCB CSU satisfies section 3.2.1.2.2.2.3 of the specific requirements of the DMCC system.

##### 4.5.12.12.2 CSU MsgsReplyCB design.

The information identified below represents the detailed design of the MsgsReplyCB CSU.

###### 4.5.12.12.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

**4.5.12.12.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

**4.5.12.12.2.3 Interrupts and signals. None****4.5.12.12.2.4 Algorithms. None****4.5.12.12.2.5 Error handling. None****4.5.12.12.2.6 Data conversion. None****4.5.12.12.2.7 Use of other elements. Other elements that are used by the MsgsReplyCB CSU:**

"msg1List" - widget id of the msg1 list

"replyFlag" - is this a reply

"replyCEOI" - reply return address

"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form

**4.5.12.12.2.8 Logic flow. Single pass.****4.5.12.12.2.9 Data structures. None****4.5.12.12.2.10 Local data files or database. None****4.5.12.12.2.11 Limitations. None****4.5.12.13 CSU MsgsReportCB**

The MsgsReportCB CSU handles the callback for the Report pushbutton. It unmanages the Msgs1 screen and brings up the Reports screen.

**4.5.12.13.1 CSU MsgsReportCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.12.13.2 CSU MsgsReportCB design.**

The information identified below represents the detailed design of the MsgsReportCB CSU.



**4.5.12.13.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.12.13.2.2 Local data elements. None****4.5.12.13.2.3 Interrupts and signals. None****4.5.12.13.2.4 Algorithms. None****4.5.12.13.2.5 Error handling. None****4.5.12.13.2.6 Data conversion. None****4.5.12.13.2.7 Use of other elements. Other elements that are used by the MsgsReportCB CSU:**

"msg1Tmi" - widget id of TMI area of msg1 form

"msg1Form" - widget id of entry area of msg1 form

**4.5.12.13.2.8 Logic flow. Single pass.****4.5.12.13.2.9 Data structures. None****4.5.12.13.2.10 Local data files or database. None****4.5.12.13.2.11 Limitations. None****4.5.13 Sub-Level CSC msgReadCB.**

The following paragraphs under this section describe the relationship of the msgReadCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.13.1 CSU MsgReadReadCB**

The MsgReadReadCB CSU handles the callback for the Read pushbutton. It brings up the Msg1 screen.

**4.5.13.1.1 CSU MsgReadReadCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.13.1.2 CSU MsgReadReadCB design.**

The information identified below represents the detailed design of the MsgReadReadCB CSU.

**4.5.13.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.13.1.2.2 Local data elements.**

"nullChar" - empty string

**4.5.13.1.2.3 Interrupts and signals. None****4.5.13.1.2.4 Algorithms. None****4.5.13.1.2.5 Error handling. None****4.5.13.1.2.6 Data conversion. None****4.5.13.1.2.7 Use of other elements. Other elements that are used by the MsgReadReadCB CSU:**

"msgReadTmi" - widget id of TMI area of msgRead form

"msgReadForm" - widget id of entry area of msgRead form

"cikText1" - widget id of CIK text 1

**4.5.13.1.2.8 Logic flow. Single pass.****4.5.13.1.2.9 Data structures. None****4.5.13.1.2.10 Local data files or database. None****4.5.13.1.2.11 Limitations. None**

#### **4.5.13.2 CSU MsgReadDeleteCB**

The MsgReadDeleteCB CSU handles the callback for the Delete button. It delete the message currently being displayed and returns to the Msgs1 screen.

##### **4.5.13.2.1 CSU MsgReadDeleteCB requirements.**

The MsgReadDeleteCB CSU satisfies section 3.2.1.2.2.3.1.2, 3.2.1.2.2.3.5.1 and 3.2.1.2.2.3.5.2 of the specific requirments of the DMCC system.

##### **4.5.13.2.2 CSU MsgReadDeleteCB design.**

The information identified below represents the detailed design of the MsgReadDeleteCB CSU.

###### **4.5.13.2.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

###### **4.5.13.2.2.2 Local data elements.**

###### **4.5.13.2.2.3 Interrupts and signals. None**

###### **4.5.13.2.2.4 Algorithms. None**

###### **4.5.13.2.2.5 Error handling. None**

###### **4.5.13.2.2.6 Data conversion. None**

###### **4.5.13.2.2.7 Use of other elements. Other elements that are used by the MsgReadDeleteCB CSU: None**

###### **4.5.13.2.2.8 Logic flow. Single pass.**

###### **4.5.13.2.2.9 Data structures. None**

###### **4.5.13.2.2.10 Local data files or database. None**

###### **4.5.13.2.2.11 Limitations. None**

#### **4.5.13.3 CSU PutupMsgRead .**

The PutupMsgRead CSU will create the MsgRead screen if necessary and will also manage the screen.

##### **4.5.13.3.1 CSU PutupMsgRead requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.13.3.2 CSU PutupMsgRead design.**

The information identified below represents the detailed design of the PutupMsgRead CSU.

###### **4.5.13.3.2.1 Input/output data elements.**

INPUTS: None

OUTPUTS: None

###### **4.5.13.3.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

###### **4.5.13.3.2.3 Interrupts and signals. None**

###### **4.5.13.3.2.4 Algorithms. None**

###### **4.5.13.3.2.5 Error handling. None**

###### **4.5.13.3.2.6 Data conversion. None**

###### **4.5.13.3.2.7 Use of other elements. Other elements that are used by the PutupMsgRead CSU:**

"msgReadTmi" - widget id of TMI area of msgRead form

"msgReadForm" - widget id of entry area of msgRead form

"smdForm" - widget id of the smd

"tmiScreen" - widget id of current TMI

"smdScreen" - widget id of current entry screen

4.5.13.3.2.8 Logic flow. Single pass.

4.5.13.3.2.9 Data structures. None

4.5.13.3.2.10 Local data files or database. None

4.5.13.3.2.11 Limitations. None

#### 4.5.13.4 CSU MsgReadReuseNIncCB.

The MsgsReuseNIncCB CSU handles the callback for the Reuse and Include pushbutton. It saves the currently displayed PDU in a global and then brings up the report screen so that a new report can be generated for inclusion with the reused message.

##### 4.5.13.4.1 CSU MsgReadReuseNIncCB requirements.

The MsgReadReuseNIncCB CSU satisfies section 3.2.1.2.2.3.1.2 and 3.2.1.2.2.3.2.3 of the specific requirements of the DMCC system.

##### 4.5.13.4.2 CSU MsgReadReuseNIncCB design.

The information identified below represents the detailed design of the MsgReadReuseNIncCB CSU.

###### 4.5.13.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.13.4.2.2 Local data elements.

"pos" - position in list highlighted

"count" - number of items highlighted

###### 4.5.13.4.2.3 Interrupts and signals. None

###### 4.5.13.4.2.4 Algorithms. None

###### 4.5.13.4.2.5 Error handling. None

###### 4.5.13.4.2.6 Data conversion. None

4.5.13.4.2.7 Use of other elements. Other elements that are used by the MsgReadReuseNIncCB CSU:

"msgReadTmi" - widget id of TMI area of msgRead form

"msgReadForm" - widget id of entry area of msgRead form

"msg1List" - widget id of msg1 list

"reusePDU" - pdu to be forwarded

"reuseFlag" - is this a forwarded message

"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.13.4.2.8 Logic flow. Single pass.

4.5.13.4.2.9 Data structures. None

4.5.13.4.2.10 Local data files or database. None

4.5.13.4.2.11 Limitations. None

#### 4.5.13.5 CSU MsgReadReuseCB

The MsgsReuseCB CSU handles the callback for the Reuse pushbutton. It saves the currently displayed PDU in a global and thing brings up the reuse report screen.

##### 4.5.13.5.1 CSU MsgReadReuseCB requirements.

The MsgReadReuseCB CSU satisfies section 3.2.1.2.2.3.1.2 and 3.2.1.2.2.3.2.2 of the specific requirments of the DMCC system.

##### 4.5.13.5.2 CSU MsgReadReuseCB design.

The information identified below represents the detailed design of the MsgReadReuseCB CSU.

###### 4.5.13.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

###### 4.5.13.5.2.2 Local data elements.

"pos" - position in list highlighted

"count" - number of items highlighted

4.5.13.5.2.3 Interrupts and signals. None

4.5.13.5.2.4 Algorithms. None

4.5.13.5.2.5 Error handling. None

4.5.13.5.2.6 Data conversion. None

4.5.13.5.2.7 Use of other elements. Other elements that are used by the MsgReadReuseCB CSU:

"msgReadTmi" - widget id of TMI area of msgRead form

"msgReadForm" - widget id of entry area of msgRead form

"msg1List" - widget id of msg1 list

"reusePDU" - pdu to be forwarded

"reuseFlag" - is this a forwarded message

"msgListIndex" - array of mappings of msg1 list entries to messages entries

4.5.13.5.2.8 Logic flow. Single pass.

4.5.13.5.2.9 Data structures. None

4.5.13.5.2.10 Local data files or database. None

4.5.13.5.2.11 Limitations. None

#### 4.5.13.6 CSU MsgReplyCB

The MsgsReplyCB callback handles the callback for the Reply pushbutton. It stores the CEOI of the sender of the displayed message and then brings up the report screen for a message to be generated to the sender of said message.

##### 4.5.13.6.1 CSU MsgReplyCB requirements.

The MsgReplyCB CSU satisfies section 3.2.1.2.2.3.1.2 of the specific requirements of the DMCC system.

##### 4.5.13.6.2 CSU MsgReplyCB design.

The information identified below represents the detailed design of the MsgReplyCB CSU.

**4.5.13.6.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.13.6.2.2 Local data elements.**

"pos" - position in list highlighted

"count" - number of items highlighted

**4.5.13.6.2.3 Interrupts and signals. None****4.5.13.6.2.4 Algorithms. None****4.5.13.6.2.5 Error handling. None****4.5.13.6.2.6 Data conversion. None****4.5.13.6.2.7 Use of other elements. Other elements that are used by the MsgReplyCB CSU:**

"msgReadTmi" - widget id of TMI area of msgRead form

"msgReadForm" - widget id of entry area of msgRead form

"msg1List" - widget id of msg1 list

"replyFlag" - is this a reply

"replyCEOI" - reply return address

**4.5.13.6.2.8 Logic flow. Single pass.****4.5.13.6.2.9 Data structures. None****4.5.13.6.2.10 Local data files or database. None****4.5.13.6.2.11 Limitations. None****4.5.14 Sub-Level CSC mtoTmi\_call.**

The following paragraphs under this section describe the relationship of the mtoTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.



#### 4.5.14.1 CSU MtoClearNRetCB

The MtoClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Mto window and remanage the Reports window.

##### 4.5.14.1.1 CSU MtoClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.14.1.2 CSU MtoClearNRetCB design.

The information identified below represents the detailed design of the MtoClearNRetCB CSU.

###### 4.5.14.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.14.1.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

###### 4.5.14.1.2.3 Interrupts and signals. None

###### 4.5.14.1.2.4 Algorithms. None

###### 4.5.14.1.2.5 Error handling. None

###### 4.5.14.1.2.6 Data conversion. None

###### 4.5.14.1.2.7 Use of other elements. Other elements that are used by the MtoClearNRetCB CSU:

"MtoTmiForm" - form widget containing Mto TMI

"MtoForm" - form widget containing Mto Entry area

"cikLabel2" - label widget

4.5.14.1.2.8 Logic flow. Single pass

4.5.14.1.2.9 Data structures. None

4.5.14.1.2.10 Local data files or database. None

4.5.14.1.2.11 Limitations. None

#### 4.5.14.2 CSU MtoSaveNRetCB

The MtoSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Mto window and remanage the Reports window.

##### 4.5.14.2.1 CSU MtoSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.14.2.2 CSU MtoSaveNRetCB design.

The information identified below represents the detailed design of the MtoSaveNRetCB CSU.

###### 4.5.14.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.14.2.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

- 4.5.14.2.2.3 Interrupts and signals. None
- 4.5.14.2.2.4 Algorithms. None
- 4.5.14.2.2.5 Error handling. None
- 4.5.14.2.2.6 Data conversion. None
- 4.5.14.2.2.7 Use of other elements. Other elements that are used by the MtoSaveNRetCB CSU:
  - "MtoTmiForm" - form widget containing Mto TMI
  - "MtoForm" - form widget containing Mto Entry area
  - "cikLabel2" - label widget
  - "cikLabel1" - label widget
- 4.5.14.2.2.8 Logic flow. Single pass
- 4.5.14.2.2.9 Data structures. None
- 4.5.14.2.2.10 Local data files or database. None
- 4.5.14.2.2.11 Limitations. None

#### **4.5.14.3 CSU MtoSndRoutCB**

The MtoSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls MtoSend with priority ROUTINE.

##### **4.5.14.3.1 CSU MtoSndRoutCB requirements**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.14.3.2 CSU MtoSndRoutCB design.**

The information identified below represents the detailed design of the MtoSndRoutCB CSU.

###### **4.5.14.3.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.14.3.2.2 Local data elements. None

4.5.14.3.2.3 Interrupts and signals. None

4.5.14.3.2.4 Algorithms. None

4.5.14.3.2.5 Error handling. None

4.5.14.3.2.6 Data conversion. None

4.5.14.3.2.7 Use of other elements. Other elements that are used by the MtoSndRoutCB CSU:

"PRIORITY\_ROUTINE" - define for priority message

4.5.14.3.2.8 Logic flow. Single pass

4.5.14.3.2.9 Data structures. None

4.5.14.3.2.10 Local data files or database. None

4.5.14.3.2.11 Limitations. None

#### 4.5.14.4 CSU MtoSend

The MtoSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldMto to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

##### 4.5.14.4.1 CSU MtoSend requirements.

The MtoSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

##### 4.5.14.4.2 CSU MtoSend design.

The information identified below represents the detailed design of the MtoSend CSU.

**4.5.14.4.2.1 Input/output data elements.****INPUTS:**

"prio" - priority of the message being sent

OUTPUTS: None

**4.5.14.4.2.2 Local data elements.**

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

"MissionStatus" - boolean; is this a Mission Status message

"RequestAdjustment" - boolean; is this a Request Adjustment message

"EnterTarget" - boolean; is this a Enter Target message

"EndMission" - boolean; is this a End Mission message

**4.5.14.4.2.3 Interrupts and signals. None****4.5.14.4.2.4 Algorithms. None****4.5.14.4.2.5 Error handling. None****4.5.14.4.2.6 Data conversion. None****4.5.14.4.2.7 Use of other elements. Other elements that are used by the MtoSend CSU:**

"MtoAdd" - information on the Mto address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call

"reuseFlag" - boolean; is this a reuse and include

**4.5.14.4.2.8 Logic flow. Single pass****4.5.14.4.2.9 Data structures. None****4.5.14.4.2.10 Local data files or database. None****4.5.14.4.2.11 Limitations. None****4.5.14.5 CSU MtoSndUrgCB**

The MtoSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls MtoSend with priority URGENT.

**4.5.14.5.1 CSU MtoSndUrgCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.14.5.2 CSU MtoSndUrgCB design.**

The information identified below represents the detailed design of the MtoSndUrgCB CSU.

**4.5.14.5.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.14.5.2.2 Local data elements. None****4.5.14.5.2.3 Interrupts and signals. None****4.5.14.5.2.4 Algorithms. None****4.5.14.5.2.5 Error handling. None****4.5.14.5.2.6 Data conversion. None****4.5.14.5.2.7 Use of other elements. Other elements that are used by the MtoSndUrgCB CSU:**

"PRIORITY\_URGENT" - define for urgent priority

**4.5.14.5.2.8 Logic flow. Single pass****4.5.14.5.2.9 Data structures. None****4.5.14.5.2.10 Local data files or database. None****4.5.14.5.2.11 Limitations. None**

#### **4.5.14.6 CSU ClearMto**

The ClearMto CSU clears all the entries of the Mto report generation screen.

##### **4.5.14.6.1 CSU ClearMto requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.14.6.2 CSU ClearMto design.**

The information identified below represents the detailed design of the ClearMto CSU.

###### **4.5.14.6.2.1 Input/output data elements.**

INPUTS:

OUTPUTS: None

###### **4.5.14.6.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

###### **4.5.14.6.2.3 Interrupts and signals. None**

###### **4.5.14.6.2.4 Algorithms. None**

###### **4.5.14.6.2.5 Error handling. None**

###### **4.5.14.6.2.6 Data conversion. None**

###### **4.5.14.6.2.7 Use of other elements. Other elements that are used by the ClearMto CSU:**

"MtoAdd" - info on the selection menu for the address

"EnemyType" - info on the selection menu for the enemy type

"EnemyActivity" - info on the selection menu for the enemy activity

"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions

"SpeedText" - widget id for speed text

"NumText" - widget id for number text

"UnitText" - widget id for unit text

"cikText1" - widget id for cik1 text

"cikText2" - widget id for cik2 text

4.5.14.6.2.8 Logic flow. Single pass

4.5.14.6.2.9 Data structures. None

4.5.14.6.2.10 Local data files or database. None

4.5.14.6.2.11 Limitations. None

#### 4.5.15 Sub-Level CSC mto\_call.

The following paragraphs under this section describe the relationship of the mto\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.15.1 CSU mtoad\_moreCB

The mtoad\_moreCB CSU handles the callback for the mtoad More pushbutton. It displays additional selectins if they exist.

##### 4.5.15.1.1 CSU mtoad\_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.15.1.2 CSU mtoad\_moreCB design.

The information identified below represents the detailed design of the mtoad\_moreCB CSU.

##### 4.5.15.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason



**OUTPUTS: .**

- 4.5.15.1.2.2 Local data elements. None
- 4.5.15.1.2.3 Interrupts and signals. None
- 4.5.15.1.2.4 Algorithms. None
- 4.5.15.1.2.5 Error handling. None
- 4.5.15.1.2.6 Data conversion. None
- 4.5.15.1.2.7 Use of other elements. Other elements that are used by the mtoad\_moreCB CSU:  
"MtoAdd" - info on the Mto address selection menu
- 4.5.15.1.2.8 Logic flow. Single pass
- 4.5.15.1.2.9 Data structures. None
- 4.5.15.1.2.10 Local data files or database. None
- 4.5.15.1.2.11 Limitations. None

**4.5.15.2 CSU mtoad\_rotateCB**

The mtoad\_rotateCB CSU handles the callback for the mtoad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.15.2.1 CSU mtoad\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.15.2.2 CSU mtoad\_rotateCB design.**

The information identified below represents the detailed design of the mtoad\_rotateCB CSU.

**4.5.15.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .****4.5.15.2.2.2 Local data elements. None****4.5.15.2.2.3 Interrupts and signals. None****4.5.15.2.2.4 Algorithms. None****4.5.15.2.2.5 Error handling. None****4.5.15.2.2.6 Data conversion. None****4.5.15.2.2.7 Use of other elements. Other elements that are used by the mtoad\_rotateCB CSU:**

"MtoAdd" - info on Mto address selection menu

**4.5.15.2.2.8 Logic flow. Single pass****4.5.15.2.2.9 Data structures. None****4.5.15.2.2.10 Local data files or database. None****4.5.15.2.2.11 Limitations. None****4.5.15.3 CSU PutupMto**

The PutupMto CSU will create the Mto report screen if necessary and will also manage the screen.

**4.5.15.3.1 CSU PutupMto requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.15.3.2 CSU PutupMto design.**

The information identified below represents the detailed design of the PutupMto CSU.

**4.5.15.3.2.1 Input/output data elements.**

INPUTS: None

OUTPUTS: None

**4.5.15.3.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

**4.5.15.3.2.3 Interrupts and signals. None****4.5.15.3.2.4 Algorithms. None****4.5.15.3.2.5 Error handling. None****4.5.15.3.2.6 Data conversion. None****4.5.15.3.2.7 Use of other elements. Other elements that are used by the PutupMto CSU:**

"MtoForm" - widget id of mto form

"MtoAdd" - info on Mto address selection menu

"addressList" - list of addresses

"MtoMoreLabel" - widget id of More label

"MtoMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply

"MtoTmiForm" widget id of mto TMI form

**4.5.15.3.2.8 Logic flow. Single pass****4.5.15.3.2.9 Data structures. None****4.5.15.3.2.10 Local data files or database. None****4.5.15.3.2.11 Limitations. None**

#### **4.5.16 Sub-Level CSC popupMessage.**

The following paragraphs under this section describe the relationship of the popupMessage Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.16.1 CSU CreateMessageBox**

The CreateMessageBox CSU creates a Popup message box with a given message and a given parent.

##### **4.5.16.1.1 CSU CreateMessageBox requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.16.1.2 CSU CreateMessageBox design.**

The information identified below represents the detailed design of the CreateMessageBox CSU.

###### **4.5.16.1.2.1 Input/output data elements.**

###### **INPUTS:**

"parent" - widget id of parent object

"messageString" - string to be displayed in the message box

###### **OUTPUTS:**

"retval" - widget id of the message box

###### **4.5.16.1.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

"popupMessage" - widget id of message box

###### **4.5.16.1.2.3 Interrupts and signals. None**

###### **4.5.16.1.2.4 Algorithms. None**

###### **4.5.16.1.2.5 Error handling. None**

4.5.16.1.2.6 Data conversion. None

4.5.16.1.2.7 Use of other elements. Other elements that are used by the CreateMessageBox CSU:  
None

4.5.16.1.2.8 Logic flow. Single pass.

4.5.16.1.2.9 Data structures. None

4.5.16.1.2.10 Local data files or database. None

4.5.16.1.2.11 Limitations. None

#### 4.5.16.2 CSU createMessage

The createMessage CSU unmanages the Cancel and Help buttons of the Message Box created by CreateMessageBox.

##### 4.5.16.2.1 CSU createMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.16.2.2 CSU createMessage design.

The information identified below represents the detailed design of the createMessage CSU.

###### 4.5.16.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.16.2.2.2 Local data elements. None

4.5.16.2.2.3 Interrupts and signals. None

4.5.16.2.2.4 Algorithms. None

4.5.16.2.2.5 Error handling. None

4.5.16.2.2.6 Data conversion. None

4.5.16.2.2.7 Use of other elements. Other elements that are used by the createMessage CSU:  
None

4.5.16.2.2.8 Logic flow. Single pass.

4.5.16.2.2.9 Data structures. None

4.5.16.2.2.10 Local data files or database. None

4.5.16.2.2.11 Limitations. None

#### 4.5.16.3 CSU okbutton

The okbutton CSU handles the callback for the OK button. It destroys the Message Box.

##### 4.5.16.3.1 CSU okbutton requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.16.3.2 CSU okbutton design.

The information identified below represents the detailed design of the okbutton CSU.

###### 4.5.16.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.16.3.2.2 Local data elements. None

- 4.5.16.3.2.3 Interrupts and signals. None
- 4.5.16.3.2.4 Algorithms. None
- 4.5.16.3.2.5 Error handling. None
- 4.5.16.3.2.6 Data conversion. None
- 4.5.16.3.2.7 Use of other elements. Other elements that are used by the okbutton CSU:  
None
- 4.5.16.3.2.8 Logic flow. Single pass.
- 4.5.16.3.2.9 Data structures. None
- 4.5.16.3.2.10 Local data files or database. None
- 4.5.16.3.2.11 Limitations. None

#### 4.5.16.4 CSU unmanageMessage

The unmanageMessage CSU handles the timer event set for three seconds and destorys the Message Box.

##### 4.5.16.4.1 CSU unmanageMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.16.4.2 CSU unmanageMessage design.

The information identified below represents the detailed design of the unmanageMessage CSU.

###### 4.5.16.4.2.1 Input/output data elements.

###### INPUTS:

"data" - widget id of widget to be destroyed

"id" - calling data

OUTPUTS: None

- 4.5.16.4.2.2 Local data elements. None
- 4.5.16.4.2.3 Interrupts and signals. None
- 4.5.16.4.2.4 Algorithms. None
- 4.5.16.4.2.5 Error handling. None
- 4.5.16.4.2.6 Data conversion. None
- 4.5.16.4.2.7 Use of other elements. Other elements that are used by the unmanageMessage CSU:  
None
- 4.5.16.4.2.8 Logic flow. Single pass.
- 4.5.16.4.2.9 Data structures. None
- 4.5.16.4.2.10 Local data files or database. None
- 4.5.16.4.2.11 Limitations. None

#### 4.5.16.5 CSU popupTransientMessage

The popupTransientMessage CSU creates a Message Box which can either be dismissed or will destroy itself after 5 seconds.

##### 4.5.16.5.1 CSU popupTransientMessage requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.16.5.2 CSU popupTransientMessage design.

The information identified below represents the detailed design of the popupTransientMessage CSU.

###### 4.5.16.5.2.1 Input/output data elements.

###### INPUTS:

"parent" - widget id of parent object

"messageString" - string to be displayed in the message box



OUTPUTS: None

**4.5.16.5.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

**4.5.16.5.2.3 Interrupts and signals. None**

**4.5.16.5.2.4 Algorithms. None**

**4.5.16.5.2.5 Error handling. None**

**4.5.16.5.2.6 Data conversion. None**

**4.5.16.5.2.7 Use of other elements. Other elements that are used by the popupTransientMessage CSU:  
None**

**4.5.16.5.2.8 Logic flow. Single pass.**

**4.5.16.5.2.9 Data structures. None**

**4.5.16.5.2.10 Local data files or database. None**

**4.5.16.5.2.11 Limitations. None**

**4.5.16.6 CSU popupMessageBox**

The popupMessageBox CSU creates a Message Box which can be dismissed.

**4.5.16.6.1 CSU popupMessageBox requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.16.6.2 CSU popupMessageBox design.**

The information identified below represents the detailed design of the popupMessageBox CSU.

**4.5.16.6.2.1 Input/output data elements.****INPUTS:**

"parent" - widget id of parent object

"messageString" - string to be displayed in the message box

**OUTPUTS:** None

**4.5.16.6.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

**4.5.16.6.2.3 Interrupts and signals. None****4.5.16.6.2.4 Algorithms. None****4.5.16.6.2.5 Error handling. None****4.5.16.6.2.6 Data conversion. None****4.5.16.6.2.7 Use of other elements. Other elements that are used by the popupMessageBox CSU:**

None

**4.5.16.6.2.8 Logic flow. Single pass.****4.5.16.6.2.9 Data structures. None****4.5.16.6.2.10 Local data files or database. None****4.5.16.6.2.11 Limitations. None****4.5.17 Sub-Level CSC reportCB.**

The following paragraphs under this section describe the relationship of the reportCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.17.1 CSU ReportMtoCB**

The ReportMtoCB CSU is the callback that brings up the MTO report screen for report entry.

**4.5.17.1.1 CSU ReportMtoCB requirements.**

The ReportMtoCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

**4.5.17.1.2 CSU ReportMtoCB design.**

The information identified below represents the detailed design of the ReportMtoCB CSU.

**4.5.17.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.17.1.2.2 Local data elements. None****4.5.17.1.2.3 Interrupts and signals. None****4.5.17.1.2.4 Algorithms. None****4.5.17.1.2.5 Error handling. None****4.5.17.1.2.6 Data conversion. None****4.5.17.1.2.7 Use of other elements. Other elements that are used by the ReportMtoCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

**4.5.17.1.2.8 Logic flow. Single pass.****4.5.17.1.2.9 Data structures. None****4.5.17.1.2.10 Local data files or database. None****4.5.17.1.2.11 Limitations. None**

#### 4.5.17.2 CSU ReportSplashCB

The ReportSplashCB CSU is the callback that brings up the Splash report screen for report entry.

##### 4.5.17.2.1 CSU ReportSplashCB requirements.

The ReportSplashCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

##### 4.5.17.2.2 CSU ReportSplashCB design.

The information identified below represents the detailed design of the ReportSplashCB CSU.

###### 4.5.17.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.17.2.2.2 Local data elements. None

###### 4.5.17.2.2.3 Interrupts and signals. None

###### 4.5.17.2.2.4 Algorithms. None

###### 4.5.17.2.2.5 Error handling. None

###### 4.5.17.2.2.6 Data conversion. None

###### 4.5.17.2.2.7 Use of other elements. Other elements that are used by the ReportSplashCB CSU:

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

###### 4.5.17.2.2.8 Logic flow. Single pass.

###### 4.5.17.2.2.9 Data structures. None

###### 4.5.17.2.2.10 Local data files or database. None

###### 4.5.17.2.2.11 Limitations. None

### **4.5.17.3 CSU ReportReqCB**

The ReportReqCB CSU is the callback that brings up the Request report screen for report entry.

#### **4.5.17.3.1 CSU ReportReqCB requirements.**

The ReportReqCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

#### **4.5.17.3.2 CSU ReportReqCB design.**

The information identified below represents the detailed design of the ReportReqCB CSU.

##### **4.5.17.3.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

##### **4.5.17.3.2.2 Local data elements. None**

##### **4.5.17.3.2.3 Interrupts and signals. None**

##### **4.5.17.3.2.4 Algorithms. None**

##### **4.5.17.3.2.5 Error handling. None**

##### **4.5.17.3.2.6 Data conversion. None**

##### **4.5.17.3.2.7 Use of other elements. Other elements that are used by the ReportReqCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

##### **4.5.17.3.2.8 Logic flow. Single pass.**

##### **4.5.17.3.2.9 Data structures. None**

##### **4.5.17.3.2.10 Local data files or database. None**

##### **4.5.17.3.2.11 Limitations. None**

#### 4.5.17.4 CSU ReportMovcmdCB

The ReportMovcmdCB CSU is the callback that brings up the Movcmd report screen for report entry.

##### 4.5.17.4.1 CSU ReportMovcmdCB requirements.

The ReportMovcmdCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

##### 4.5.17.4.2 CSU ReportMovcmdCB design.

The information identified below represents the detailed design of the ReportMovcmdCB CSU.

###### 4.5.17.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

###### 4.5.17.4.2.2 Local data elements. None

###### 4.5.17.4.2.3 Interrupts and signals. None

###### 4.5.17.4.2.4 Algorithms. None

###### 4.5.17.4.2.5 Error handling. None

###### 4.5.17.4.2.6 Data conversion. None

###### 4.5.17.4.2.7 Use of other elements. Other elements that are used by the ReportMovcmdCB CSU:

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

###### 4.5.17.4.2.8 Logic flow. Single pass.

###### 4.5.17.4.2.9 Data structures. None

###### 4.5.17.4.2.10 Local data files or database. None

###### 4.5.17.4.2.11 Limitations. None

**4.5.17.5 CSU ReportShotCB**

The ReportShotCB CSU is the callback that brings up the Shot report screen for report entry.

**4.5.17.5.1 CSU ReportShotCB requirements.**

The ReportShotCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

**4.5.17.5.2 CSU ReportShotCB design.**

The information identified below represents the detailed design of the ReportShotCB CSU.

**4.5.17.5.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.17.5.2.2 Local data elements. None****4.5.17.5.2.3 Interrupts and signals. None****4.5.17.5.2.4 Algorithms. None****4.5.17.5.2.5 Error handling. None****4.5.17.5.2.6 Data conversion. None****4.5.17.5.2.7 Use of other elements. Other elements that are used by the ReportShotCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

**4.5.17.5.2.8 Logic flow. Single pass.****4.5.17.5.2.9 Data structures. None****4.5.17.5.2.10 Local data files or database. None****4.5.17.5.2.11 Limitations. None**

#### 4.5.17.6 CSU ReportSpotCB

The ReportSpotCB CSU is the callback that brings up the Spot report screen for report entry.

##### 4.5.17.6.1 CSU ReportSpotCB requirements.

The ReportSpotCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

##### 4.5.17.6.2 CSU ReportSpotCB design.

The information identified below represents the detailed design of the ReportSpotCB CSU.

###### 4.5.17.6.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.17.6.2.2 Local data elements. None

###### 4.5.17.6.2.3 Interrupts and signals. None

###### 4.5.17.6.2.4 Algorithms. None

###### 4.5.17.6.2.5 Error handling. None

###### 4.5.17.6.2.6 Data conversion. None

###### 4.5.17.6.2.7 Use of other elements. Other elements that are used by the ReportSpotCB CSU:

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

###### 4.5.17.6.2.8 Logic flow. Single pass.

###### 4.5.17.6.2.9 Data structures. None

###### 4.5.17.6.2.10 Local data files or database. None

###### 4.5.17.6.2.11 Limitations. None



**4.5.17.7 CSU ReportFreeTxtCB**

The ReportFreeTxtCB CSU is the callback that brings up the Free Text report screen for report entry.

**4.5.17.7.1 CSU ReportFreeTxtCB requirements.**

The ReportFreeTxtCB CSU satisfies section 3.2.1.2.2.3.2.2 of the specific requirements of the DMCC system.

**4.5.17.7.2 CSU ReportFreeTxtCB design.**

The information identified below represents the detailed design of the ReportFreeTxtCB CSU.

**4.5.17.7.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.17.7.2.2 Local data elements. None****4.5.17.7.2.3 Interrupts and signals. None****4.5.17.7.2.4 Algorithms. None****4.5.17.7.2.5 Error handling. None****4.5.17.7.2.6 Data conversion. None****4.5.17.7.2.7 Use of other elements. Other elements that are used by the ReportFreeTxtCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

**4.5.17.7.2.8 Logic flow. Single pass.****4.5.17.7.2.9 Data structures. None****4.5.17.7.2.10 Local data files or database. None****4.5.17.7.2.11 Limitations. None**

**4.5.17.8 CSU ReportSaveExitCB****4.5.17.8.1 CSU ReportSaveExitCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.17.8.2 CSU ReportSaveExitCB design.**

The information identified below represents the detailed design of the ReportSaveExitCB CSU.

**4.5.17.8.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.17.8.2.2 Local data elements. None****4.5.17.8.2.3 Interrupts and signals. None****4.5.17.8.2.4 Algorithms. None****4.5.17.8.2.5 Error handling. None****4.5.17.8.2.6 Data conversion. None****4.5.17.8.2.7 Use of other elements. Other elements that are used by the ReportSaveExitCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

"replyFlag" - is this a reply

"reuseFlag" - is this a forwarded message

**4.5.17.8.2.8 Logic flow. Single pass.****4.5.17.8.2.9 Data structures. None****4.5.17.8.2.10 Local data files or database. None**

**4.5.17.8.2.11 Limitations. None****4.5.17.9 CSU ReportMsgsCB****4.5.17.9.1 CSU ReportMsgsCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.17.9.2 CSU ReportMsgsCB design.**

The information identified below represents the detailed design of the ReportMsgsCB CSU.

**4.5.17.9.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.17.9.2.2 Local data elements. None****4.5.17.9.2.3 Interrupts and signals. None****4.5.17.9.2.4 Algorithms. None****4.5.17.9.2.5 Error handling. None****4.5.17.9.2.6 Data conversion. None****4.5.17.9.2.7 Use of other elements. Other elements that are used by the ReportMsgsCB CSU:**

"reportTmi" - widget id for the report window TMI area

"reportForm" - widget id for the report window entry area

"replyFlag" - is this a reply

"reuseFlag" - is this a forwarded message

**4.5.17.9.2.8 Logic flow. Single pass.**

4.5.17.9.2.9 Data structures. None

4.5.17.9.2.10 Local data files or database. None

4.5.17.9.2.11 Limitations. None

#### 4.5.17.10 CSU PutupReport

The PutupReport CSU is the callback that brings up the MTO report screen for report entry.

##### 4.5.17.10.1 CSU PutupReport requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.17.10.2 CSU PutupReport design.

The information identified below represents the detailed design of the PutupReport CSU.

###### 4.5.17.10.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.17.10.2.2 Local data elements. None

4.5.17.10.2.3 Interrupts and signals. None

4.5.17.10.2.4 Algorithms. None

4.5.17.10.2.5 Error handling. None

4.5.17.10.2.6 Data conversion. None

4.5.17.10.2.7 Use of other elements. Other elements that are used by the PutupReport CSU:

"reportTmi" - widgit id for the report window TMI area  
"reportForm" - widgit id for the report window entry area  
"tmiScreen" - widget id of current TMI  
"smdScreen" - widget id of current entry screen

4.5.17.10.2.8 Logic flow. Single pass.

4.5.17.10.2.9 Data structures. None

4.5.17.10.2.10 Local data files or database. None

4.5.17.10.2.11 Limitations. None

#### **4.5.18 Sub-Level CSC reqtTmi\_call.**

The following paragraphs under this section describe the relationship of the reqtTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.18.1 CSU ReqClearNRetCB**

The ReqClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Req window and remanage the Reports window.

##### **4.5.18.1.1 CSU ReqClearNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.18.1.2 CSU ReqClearNRetCB design.**

The information identified below represents the detailed design of the ReqClearNRetCB CSU.

##### **4.5.18.1.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.18.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.18.1.2.3 Interrupts and signals. None****4.5.18.1.2.4 Algorithms. None****4.5.18.1.2.5 Error handling. None****4.5.18.1.2.6 Data conversion. None****4.5.18.1.2.7 Use of other elements. Other elements that are used by the ReqtClearNRetCB CSU:**

"ReqTmiForm" - form widget containing Reqt TMI

"ReqForm" - form widget containing Reqt Entry area

"cikLabel2" - label widget

**4.5.18.1.2.8 Logic flow. Single pass****4.5.18.1.2.9 Data structures. None****4.5.18.1.2.10 Local data files or database. None****4.5.18.1.2.11 Limitations. None****4.5.18.2 CSU ReqtSaveNRetCB**

The ReqtSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Reqt window and remanage the Reports window.

**4.5.18.2.1 CSU ReqtSaveNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.18.2.2 CSU ReqSaveNRetCB design.**

The information identified below represents the detailed design of the ReqSaveNRetCB CSU.

**4.5.18.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.18.2.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

**4.5.18.2.2.3 Interrupts and signals. None****4.5.18.2.2.4 Algorithms. None****4.5.18.2.2.5 Error handling. None****4.5.18.2.2.6 Data conversion. None****4.5.18.2.2.7 Use of other elements. Other elements that are used by the ReqSaveNRetCB CSU:**

"ReqTmiForm" - form widget containing Req TMI

"ReqForm" - form widget containing Req Entry area

"cikLabel2" - label widget

"cikLabel1" - label widget

**4.5.18.2.2.8 Logic flow. Single pass****4.5.18.2.2.9 Data structures. None****4.5.18.2.2.10 Local data files or database. None****4.5.18.2.2.11 Limitations. None**

#### 4.5.18.3 CSU ReqSndRoutCB

The ReqSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls ReqSend with priority ROUTINE.

##### 4.5.18.3.1 CSU ReqSndRoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.18.3.2 CSU ReqSndRoutCB design.

The information identified below represents the detailed design of the ReqSndRoutCB CSU.

###### 4.5.18.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.18.3.2.2 Local data elements. None

###### 4.5.18.3.2.3 Interrupts and signals. None

###### 4.5.18.3.2.4 Algorithms. None

###### 4.5.18.3.2.5 Error handling. None

###### 4.5.18.3.2.6 Data conversion. None

###### 4.5.18.3.2.7 Use of other elements. Other elements that are used by the ReqSndRoutCB CSU:

"PRIORITY\_ROUTINE" - define for priority message

###### 4.5.18.3.2.8 Logic flow. Single pass

###### 4.5.18.3.2.9 Data structures. None

###### 4.5.18.3.2.10 Local data files or database. None



**4.5.18.3.2.11 Limitations. None****4.5.18.4 CSU ReqSend**

The ReqSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldReq to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

**4.5.18.4.1 CSU ReqSend requirements**

The ReqSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

**4.5.18.4.2 CSU ReqSend design.**

The information identified below represents the detailed design of the ReqSend CSU.

**4.5.18.4.2.1 Input/output data elements.****INPUTS:**

"prio" - priority of the message being sent

**OUTPUTS: None****4.5.18.4.2.2 Local data elements.**

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

"reportType" - type of report requested

"reconType" - type of recon

**4.5.18.4.2.3 Interrupts and signals. None****4.5.18.4.2.4 Algorithms. None****4.5.18.4.2.5 Error handling. None****4.5.18.4.2.6 Data conversion. None****4.5.18.4.2.7 Use of other elements. Other elements that are used by the ReqSend CSU:**

"ReqAdd" - information on the Req address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call  
"Type" - information on the Type selection menu  
"Recon" - information on the Recon selection menu  
"reuseFlag" - boolean; is this a reuse and include

4.5.18.4.2.8 Logic flow. Single pass

4.5.18.4.2.9 Data structures. None

4.5.18.4.2.10 Local data files or database. None

4.5.18.4.2.11 Limitations. None

#### 4.5.18.5 CSU ReqTsndUrgCB

The ReqTsndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls ReqTsend with priority URGENT.

##### 4.5.18.5.1 CSU ReqTsndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.18.5.2 CSU ReqTsndUrgCB design.

The information identified below represents the detailed design of the ReqTsndUrgCB CSU.

###### 4.5.18.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.18.5.2.2 Local data elements. None

4.5.18.5.2.3 Interrupts and signals. None

4.5.18.5.2.4 Algorithms. None

4.5.18.5.2.5 Error handling. None

4.5.18.5.2.6 Data conversion. None

4.5.18.5.2.7 Use of other elements. Other elements that are used by the ReqTsndUrgCB CSU:  
"PRIORITY\_URGENT" - define for urgent priority

4.5.18.5.2.8 Logic flow. Single pass

4.5.18.5.2.9 Data structures. None

4.5.18.5.2.10 Local data files or database. None

4.5.18.5.2.11 Limitations. None

#### 4.5.18.6 CSU ClearReqT

The ClearReqT CSU clears all the entries of the ReqT report generation screen.

##### 4.5.18.6.1 CSU ClearReqT requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.18.6.2 CSU ClearReqT design.

The information identified below represents the detailed design of the ClearReqT CSU.

###### 4.5.18.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

###### 4.5.18.6.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

- 4.5.18.6.2.3 Interrupts and signals. None
- 4.5.18.6.2.4 Algorithms. None
- 4.5.18.6.2.5 Error handling. None
- 4.5.18.6.2.6 Data conversion. None
- 4.5.18.6.2.7 Use of other elements. Other elements that are used by the ClearReq CSU:
  - "ReqAdd" - info on the selection menu for the address
  - "EnemyType" - info on the selection menu for the enemy type
  - "EnemyActivity" - info on the selection menu for the enemy activity
  - "Direc" - info on the selection menu for the direction
  - "ObsInt" - info on the selection menu for the observer intentions
  - "SpeedText" - widget id for speed text
  - "NumText" - widget id for number text
  - "UnitText" - widget id for unit text
  - "cikText1" - widget id for cik1 text
  - "cikText2" - widget id for cik2 text
- 4.5.18.6.2.8 Logic flow. Single pass
- 4.5.18.6.2.9 Data structures. None
- 4.5.18.6.2.10 Local data files or database. None
- 4.5.18.6.2.11 Limitations. None

#### 4.5.19 Sub-Level CSC reqt\_call.

The following paragraphs under this section describe the relationship of the reqt\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.19.1 CSU type\_moreCB

The typead\_moreCB CSU handles the callback for the typead More pushbutton. It displays additional selectins if they exist.

**4.5.19.1.1 CSU type\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.19.1.2 CSU type\_moreCB design.**

The information identified below represents the detailed design of the type\_moreCB CSU.

**4.5.19.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.19.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.19.1.2.3 Interrupts and signals. None****4.5.19.1.2.4 Algorithms. None****4.5.19.1.2.5 Error handling. None****4.5.19.1.2.6 Data conversion. None****4.5.19.1.2.7 Use of other elements. Other elements that are used by the type\_moreCB CSU:**

"Type" - info on Enemy Activity selection menu

**4.5.19.1.2.8 Logic flow. Single pass****4.5.19.1.2.9 Data structures. None****4.5.19.1.2.10 Local data files or database. None**

**4.5.19.1.2.11 Limitations. None****4.5.19.2 CSU type\_rotateCB**

The typead\_rotateCB CSU handles the callback for the typead Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.19.2.1 CSU type\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.19.2.2 CSU type\_rotateCB design.**

The information identified below represents the detailed design of the type\_rotateCB CSU.

**4.5.19.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .****4.5.19.2.2.2 Local data elements. None****4.5.19.2.2.3 Interrupts and signals. None****4.5.19.2.2.4 Algorithms. None****4.5.19.2.2.5 Error handling. None****4.5.19.2.2.6 Data conversion. None****4.5.19.2.2.7 Use of other elements. Other elements that are used by the type\_rotateCB CSU:**

"Type" - info on the enemy activity selection menu

**4.5.19.2.2.8 Logic flow. Single pass**

4.5.19.2.2.9 Data structures. None

4.5.19.2.2.10 Local data files or database. None

4.5.19.2.2.11 Limitations. None

#### 4.5.19.3 CSU recon\_rotateCB

The reconad\_rotateCB CSU handles the callback for the reconad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.19.3.1 CSU recon\_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.19.3.2 CSU recon\_rotateCB design.

The information identified below represents the detailed design of the recon\_rotateCB CSU.

###### 4.5.19.3.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.19.3.2.2 Local data elements. None

4.5.19.3.2.3 Interrupts and signals. None

4.5.19.3.2.4 Algorithms. None

4.5.19.3.2.5 Error handling. None

4.5.19.3.2.6 Data conversion. None

4.5.19.3.2.7 Use of other elements. Other elements that are used by the recon\_rotateCB CSU:

"Recon" - info on the enemy activity selection menu

4.5.19.3.2.8 Logic flow. Single pass

4.5.19.3.2.9 Data structures. None

4.5.19.3.2.10 Local data files or database. None

4.5.19.3.2.11 Limitations. None

#### 4.5.19.4 CSU reqtad\_moreCB

The reqtad\_moreCB CSU handles the callback for the reqtad More pushbutton. It displays additional selectins if they exist.

##### 4.5.19.4.1 CSU reqtad\_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.19.4.2 CSU reqtad\_moreCB design.

The information identified below represents the detailed design of the reqtad\_moreCB CSU.

###### 4.5.19.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.19.4.2.2 Local data elements. None

4.5.19.4.2.3 Interrupts and signals. None

4.5.19.4.2.4 Algorithms. None

4.5.19.4.2.5 Error handling. None



4.5.19.4.2.6 Data conversion. None

4.5.19.4.2.7 Use of other elements. Other elements that are used by the reqtd\_moreCB CSU:

"ReqtdAdd" - info on the Reqtd address selection menu

4.5.19.4.2.8 Logic flow. Single pass

4.5.19.4.2.9 Data structures. None

4.5.19.4.2.10 Local data files or database. None

4.5.19.4.2.11 Limitations. None

#### 4.5.19.5 CSU reqtd\_rotateCB

The reqtd\_rotateCB CSU handles the callback for the reqtd Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.19.5.1 CSU reqtd\_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.19.5.2 CSU reqtd\_rotateCB design.

The information identified below represents the detailed design of the reqtd\_rotateCB CSU.

###### 4.5.19.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.19.5.2.2 Local data elements. None

4.5.19.5.2.3 Interrupts and signals. None

4.5.19.5.2.4 Algorithms. None

4.5.19.5.2.5 Error handling. None

4.5.19.5.2.6 Data conversion. None

4.5.19.5.2.7 Use of other elements. Other elements that are used by the reqtad\_rotateCB CSU:

"ReqtAdd" - info on Reqt address selection menu

4.5.19.5.2.8 Logic flow. Single pass

4.5.19.5.2.9 Data structures. None

4.5.19.5.2.10 Local data files or database. None

4.5.19.5.2.11 Limitations. None

#### 4.5.19.6 CSU PutupReqt

The PutupReqt CSU will create the Reqt report screen if necessary and will also manage the screen.

##### 4.5.19.6.1 CSU PutupReqt requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.19.6.2 CSU PutupReqt design.

The information identified below represents the detailed design of the PutupReqt CSU.

###### 4.5.19.6.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.19.6.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

4.5.19.6.2.3 Interrupts and signals. None

4.5.19.6.2.4 Algorithms. None

4.5.19.6.2.5 Error handling. None

4.5.19.6.2.6 Data conversion. None

4.5.19.6.2.7 Use of other elements. Other elements that are used by the PutupReq CSU:

"ReqForm" - widget id of reqt form

"ReqAdd" - info on Req address selection menu

"addressList" - list of addresses

"ReqMoreLabel" - widget id of More label

"ReqMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply

"ReqTmiForm" widget id of reqt TMI form

4.5.19.6.2.8 Logic flow. Single pass

4.5.19.6.2.9 Data structures. None

4.5.19.6.2.10 Local data files or database. None

4.5.19.6.2.11 Limitations. None

#### 4.5.20 Sub-Level CSC reuseTmi\_call.

The following paragraphs under this section describe the relationship of the reuseTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.20.1 CSU ReuseClearNRetCB

The ReuseClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Reuse window and remanage the Reports window.

**4.5.20.1.1 CSU ReuseClearNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.20.1.2 CSU ReuseClearNRetCB design.**

The information identified below represents the detailed design of the ReuseClearNRetCB CSU.

**4.5.20.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.20.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.20.1.2.3 Interrupts and signals. None****4.5.20.1.2.4 Algorithms. None****4.5.20.1.2.5 Error handling. None****4.5.20.1.2.6 Data conversion. None****4.5.20.1.2.7 Use of other elements. Other elements that are used by the ReuseClearNRetCB CSU:**

"ReuseTmiForm" - form widget containing Reuse TMI

"ReuseForm" - form widget containing Reuse Entry area

"cikLabel2" - label widget

**4.5.20.1.2.8 Logic flow. Single pass****4.5.20.1.2.9 Data structures. None**

4.5.20.1.2.10 Local data files or database. None

4.5.20.1.2.11 Limitations. None

#### 4.5.20.2 CSU ReuseSendCB

The ReuseSendCB CSU handles the callback from the Send pushbutton. It forwards the selected message to the selected address.

##### 4.5.20.2.1 CSU ReuseSend requirements.

The ReuseSend CSU satisfies section 3.2.1.2.2.3.3.1 of the specific requirements of the DMCC system.

##### 4.5.20.2.2 CSU ReuseSend design.

The information identified below represents the detailed design of the ReuseSend CSU.

###### 4.5.20.2.2.1 Input/output data elements.

###### INPUTS:

"prio" - priority of the message being sent

OUTPUTS: None

###### 4.5.20.2.2.2 Local data elements.

"targetCEOI" - name of user message is being sent to

###### 4.5.20.2.2.3 Interrupts and signals. None

###### 4.5.20.2.2.4 Algorithms. None

###### 4.5.20.2.2.5 Error handling. None

###### 4.5.20.2.2.6 Data conversion. None

###### 4.5.20.2.2.7 Use of other elements. Other elements that are used by the ReuseSend CSU:

"ReuseAdd" - information on the Reuse address selection menu

"reusePDU" - PDU to be forwarded

###### 4.5.20.2.2.8 Logic flow. Single pass

###### 4.5.20.2.2.9 Data structures. None

4.5.20.2.2.10 Local data files or database. None

4.5.20.2.2.11 Limitations. None

#### 4.5.20.3 CSU ClearReuse

The ClearReuse CSU clears all the entries of the Reuse report generation screen.

##### 4.5.20.3.1 CSU ClearReuse requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.20.3.2 CSU ClearReuse design.

The information identified below represents the detailed design of the ClearReuse CSU.

###### 4.5.20.3.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

###### 4.5.20.3.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

###### 4.5.20.3.2.3 Interrupts and signals. None

###### 4.5.20.3.2.4 Algorithms. None

###### 4.5.20.3.2.5 Error handling. None

###### 4.5.20.3.2.6 Data conversion. None

**4.5.20.3.2.7 Use of other elements.** Other elements that are used by the ClearReuse CSU:

"ReuseAdd" - info on the selection menu for the address

"EnemyType" - info on the selection menu for the enemy type

"EnemyActivity" - info on the selection menu for the enemy activity

"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions

"SpeedText" - widget id for speed text

"NumText" - widget id for number text

"UnitText" - widget id for unit text

"cikText1" - widget id for cik1 text

"cikText2" - widget id for cik2 text

**4.5.20.3.2.8 Logic flow.** Single pass

**4.5.20.3.2.9 Data structures.** None

**4.5.20.3.2.10 Local data files or database.** None

**4.5.20.3.2.11 Limitations.** None

**4.5.21 Sub-Level CSC reuse\_call.**

The following paragraphs under this section describe the relationship of the reuse\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

**4.5.21.1 CSU reusead\_moreCB**

The reusead\_moreCB CSU handles the callback for the reusead More pushbutton. It displays additional selectins if they exist.

**4.5.21.1.1 CSU reusead\_moreCB requirements**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.21.1.2 CSU reusead\_moreCB design.**

The information identified below represents the detailed design of the reusead\_moreCB CSU.

**4.5.21.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.21.1.2.2 Local data elements. None**

**4.5.21.1.2.3 Interrupts and signals. None**

**4.5.21.1.2.4 Algorithms. None**

**4.5.21.1.2.5 Error handling. None**

**4.5.21.1.2.6 Data conversion. None**

**4.5.21.1.2.7 Use of other elements. Other elements that are used by the reusead\_moreCB CSU:**

"ReuseAdd" - info on the Reuse address selection menu

**4.5.21.1.2.8 Logic flow. Single pass**

**4.5.21.1.2.9 Data structures. None**

**4.5.21.1.2.10 Local data files or database. None**

**4.5.21.1.2.11 Limitations. None**

**4.5.21.2 CSU reusead\_rotateCB**

The reusead\_rotateCB CSU handles the callback for the reusead Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.21.2.1 CSU reusead\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.



**4.5.21.2.2 CSU reusead\_rotateCB design.**

The information identified below represents the detailed design of the reusead\_rotateCB CSU.

**4.5.21.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.21.2.2.2 Local data elements. None**

**4.5.21.2.2.3 Interrupts and signals. None**

**4.5.21.2.2.4 Algorithms. None**

**4.5.21.2.2.5 Error handling. None**

**4.5.21.2.2.6 Data conversion. None**

**4.5.21.2.2.7 Use of other elements. Other elements that are used by the reusead\_rotateCB CSU:**

"ReuseAdd" - info on Reuse address selection menu

**4.5.21.2.2.8 Logic flow. Single pass**

**4.5.21.2.2.9 Data structures. None**

**4.5.21.2.2.10 Local data files or database. None**

**4.5.21.2.2.11 Limitations. None**

**4.5.21.3 CSU PutupReuse**

The PutupReuse CSU will create the Reuse report screen if necessary and will also manage the screen.

#### 4.5.21.3.1 CSU PutupReuse requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.5.21.3.2 CSU PutupReuse design.

The information identified below represents the detailed design of the PutupReuse CSU.

##### 4.5.21.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

##### 4.5.21.3.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

##### 4.5.21.3.2.3 Interrupts and signals. None

##### 4.5.21.3.2.4 Algorithms. None

##### 4.5.21.3.2.5 Error handling. None

##### 4.5.21.3.2.6 Data conversion. None

##### 4.5.21.3.2.7 Use of other elements. Other elements that are used by the PutupReuse CSU:

"ReuseForm" - widget id of reuse form

"ReuseAdd" - info on Reuse address selection menu

"addressList" - list of addresses

"ReuseMoreLabel" - widget id of More label

"ReuseMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply

"ReuseTmiForm" widget id of reuse TMI form

##### 4.5.21.3.2.8 Logic flow. Single pass

4.5.21.3.2.9 Data structures. None

4.5.21.3.2.10 Local data files or database. None

4.5.21.3.2.11 Limitations. None

#### **4.5.22 Sub-Level CSC shotTmi\_call.**

The following paragraphs under this section describe the relationship of the shotTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.22.1 CSU ShotClearNRetCB**

The ShotClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Shot window and remanage the Reports window.

##### **4.5.22.1.1 CSU ShotClearNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.22.1.2 CSU ShotClearNRetCB design.**

The information identified below represents the detailed design of the ShotClearNRetCB CSU.

##### **4.5.22.1.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

##### **4.5.22.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

4.5.22.1.2.3 Interrupts and signals. None

4.5.22.1.2.4 Algorithms. None

4.5.22.1.2.5 Error handling. None

4.5.22.1.2.6 Data conversion. None

4.5.22.1.2.7 Use of other elements. Other elements that are used by the ShotClearNRetCB CSU:

"ShotTmiForm" - form widget containing Shot TMI

"ShotForm" - form widget containing Shot Entry area

"cikLabel2" - label widget

4.5.22.1.2.8 Logic flow. Single pass

4.5.22.1.2.9 Data structures. None

4.5.22.1.2.10 Local data files or database. None

4.5.22.1.2.11 Limitations. None

#### 4.5.22.2 CSU ShotSaveNRetCB

The ShotSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Shot window and remanage the Reports window.

##### 4.5.22.2.1 CSU ShotSaveNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.22.2.2 CSU ShotSaveNRetCB design.

The information identified below represents the detailed design of the ShotSaveNRetCB CSU.

**4.5.22.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS:** None

**4.5.22.2.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

**4.5.22.2.2.3 Interrupts and signals. None****4.5.22.2.2.4 Algorithms. None****4.5.22.2.2.5 Error handling. None****4.5.22.2.2.6 Data conversion. None****4.5.22.2.2.7 Use of other elements. Other elements that are used by the ShotSaveNRetCB CSU:**

"ShotTmiForm" - form widget containing Shot TMI

"ShotForm" - form widget containing Shot Entry area

"cikLabel2" - label widget

"cikLabel1" - label widget

**4.5.22.2.2.8 Logic flow. Single pass****4.5.22.2.2.9 Data structures. None****4.5.22.2.2.10 Local data files or database. None****4.5.22.2.2.11 Limitations. None****4.5.22.3 CSU ShotSndRoutCB**

The ShotSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls ShotSend with priority ROUTINE.

**4.5.22.3.1 CSU ShotSndRoutCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.22.3.2 CSU ShotSndRoutCB design.**

The information identified below represents the detailed design of the ShotSndRoutCB CSU.

**4.5.22.3.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.22.3.2.2 Local data elements. None****4.5.22.3.2.3 Interrupts and signals. None****4.5.22.3.2.4 Algorithms. None****4.5.22.3.2.5 Error handling. None****4.5.22.3.2.6 Data conversion. None****4.5.22.3.2.7 Use of other elements. Other elements that are used by the ShotSndRoutCB CSU:**

"PRIORITY\_ROUTINE" - define for priority message

**4.5.22.3.2.8 Logic flow. Single pass****4.5.22.3.2.9 Data structures. None****4.5.22.3.2.10 Local data files or database. None****4.5.22.3.2.11 Limitations. None**

#### 4.5.22.4 CSU ShotSend

The ShotSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldShot to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

##### 4.5.22.4.1 CSU ShotSend requirements.

The ShotSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

##### 4.5.22.4.2 CSU ShotSend design.

The information identified below represents the detailed design of the ShotSend CSU.

###### 4.5.22.4.2.1 Input/output data elements.

###### INPUTS:

"prio" - priority of the message being sent

###### OUTPUTS: None

###### 4.5.22.4.2.2 Local data elements.

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

###### 4.5.22.4.2.3 Interrupts and signals. None

###### 4.5.22.4.2.4 Algorithms. None

###### 4.5.22.4.2.5 Error handling. None

###### 4.5.22.4.2.6 Data conversion. None

###### 4.5.22.4.2.7 Use of other elements. Other elements that are used by the ShotSend CSU:

"ShotAdd" - information on the Shot address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call

"reuseFlag" - boolean; is this a reuse and include

###### 4.5.22.4.2.8 Logic flow. Single pass

4.5.22.4.2.9 Data structures. None

4.5.22.4.2.10 Local data files or database. None

4.5.22.4.2.11 Limitations. None

#### 4.5.22.5 CSU ShotSndUrgCB

The ShotSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls ShotSend with priority URGENT.

##### 4.5.22.5.1 CSU ShotSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.22.5.2 CSU ShotSndUrgCB design.

The information identified below represents the detailed design of the ShotSndUrgCB CSU.

###### 4.5.22.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.22.5.2.2 Local data elements. None

4.5.22.5.2.3 Interrupts and signals. None

4.5.22.5.2.4 Algorithms. None

4.5.22.5.2.5 Error handling. None

4.5.22.5.2.6 Data conversion. None

4.5.22.5.2.7 Use of other elements. Other elements that are used by the ShotSndUrgCB CSU:

"PRIORITY\_URGENT" - define for urgent priority



4.5.22.5.2.8 Logic flow. Single pass

4.5.22.5.2.9 Data structures. None

4.5.22.5.2.10 Local data files or database. None

4.5.22.5.2.11 Limitations. None

#### 4.5.22.6 CSU ClearShot

The ClearShot CSU clears all the entries of the Shot report generation screen.

##### 4.5.22.6.1 CSU ClearShot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.22.6.2 CSU ClearShot design.

The information identified below represents the detailed design of the ClearShot CSU.

###### 4.5.22.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

###### 4.5.22.6.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

###### 4.5.22.6.2.3 Interrupts and signals. None

###### 4.5.22.6.2.4 Algorithms. None

###### 4.5.22.6.2.5 Error handling. None

4.5.22.6.2.6 Data conversion. None

4.5.22.6.2.7 Use of other elements. Other elements that are used by the ClearShot CSU:

"ShotAdd" - info on the selection menu for the address

"EnemyType" - info on the selection menu for the enemy type

"EnemyActivity" - info on the selection menu for the enemy activity

"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions

"SpeedText" - widget id for speed text

"NumText" - widget id for number text

"UnitText" - widget id for unit text

"cikText1"- widget id for cik1 text

"cikText2"- widget id for cik2 text

4.5.22.6.2.8 Logic flow. Single pass

4.5.22.6.2.9 Data structures. None

4.5.22.6.2.10 Local data files or database. None

4.5.22.6.2.11 Limitations. None

4.5.23 Sub-Level CSC shot\_call.

The following paragraphs under this section describe the relationship of the shot\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

4.5.23.1 CSU shotad\_moreCB

The shotad\_moreCB CSU handles the callback for the shotad More pushbutton. It displays additional selectins if they exist.

4.5.23.1.1 CSU shotad\_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.23.1.2 CSU shotad\_moreCB design.**

The information identified below represents the detailed design of the shotad\_moreCB CSU.

**4.5.23.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.23.1.2.2 Local data elements.** None

**4.5.23.1.2.3 Interrupts and signals.** None

**4.5.23.1.2.4 Algorithms.** None

**4.5.23.1.2.5 Error handling.** None

**4.5.23.1.2.6 Data conversion.** None

**4.5.23.1.2.7 Use of other elements.** Other elements that are used by the shotad\_moreCB CSU:

"ShotAdd" - info on the Shot address selection menu

**4.5.23.1.2.8 Logic flow.** Single pass

**4.5.23.1.2.9 Data structures.** None

**4.5.23.1.2.10 Local data files or database.** None

**4.5.23.1.2.11 Limitations.** None

**4.5.23.2 CSU shotad\_rotateCB**

The shotad\_rotateCB CSU handles the callback for the shotad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.23.2.1 CSU shotad\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.23.2.2 CSU shotad\_rotateCB design.**

The information identified below represents the detailed design of the shotad\_rotateCB CSU.

**4.5.23.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

4.5.23.2.2.2 Local data elements. None

4.5.23.2.2.3 Interrupts and signals. None

4.5.23.2.2.4 Algorithms. None

4.5.23.2.2.5 Error handling. None

4.5.23.2.2.6 Data conversion. None

4.5.23.2.2.7 Use of other elements. Other elements that are used by the shotad\_rotateCB CSU:

"ShotAdd" - info on Shot address selection menu

4.5.23.2.2.8 Logic flow. Single pass

4.5.23.2.2.9 Data structures. None

4.5.23.2.2.10 Local data files or database. None

4.5.23.2.2.11 Limitations. None

#### 4.5.23.3 CSU PutupShot

The PutupShot CSU will create the Shot report screen if necessary and will also manage the screen.

##### 4.5.23.3.1 CSU PutupShot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.23.3.2 CSU PutupShot design.

The information identified below represents the detailed design of the PutupShot CSU.

###### 4.5.23.3.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.23.3.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

###### 4.5.23.3.2.3 Interrupts and signals. None

###### 4.5.23.3.2.4 Algorithms. None

###### 4.5.23.3.2.5 Error handling. None

###### 4.5.23.3.2.6 Data conversion. None

###### 4.5.23.3.2.7 Use of other elements. Other elements that are used by the PutupShot CSU:

"ShotForm" - widget id of shot form

"ShotAdd" - info on Shot address selection menu

"addressList" - list of addresses

"ShotMoreLabel" - widget id of More label

"ShotMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply  
"ShotTmiForm" widget id of shot TMI form

4.5.23.3.2.8 Logic flow. Single pass

4.5.23.3.2.9 Data structures. None

4.5.23.3.2.10 Local data files or database. None

4.5.23.3.2.11 Limitations. None

#### 4.5.24 Sub-Level CSC splashTmi\_call.

The following paragraphs under this section describe the relationship of the splashTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.24.1 CSU SplashClearNRetCB

The SplashClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Splash window and remanage the Reports window.

##### 4.5.24.1.1 CSU SplashClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.24.1.2 CSU SplashClearNRetCB design.

The information identified below represents the detailed design of the SplashClearNRetCB CSU.

##### 4.5.24.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

**4.5.24.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.24.1.2.3 Interrupts and signals. None****4.5.24.1.2.4 Algorithms. None****4.5.24.1.2.5 Error handling. None****4.5.24.1.2.6 Data conversion. None****4.5.24.1.2.7 Use of other elements. Other elements that are used by the SplashClearNRetCB CSU:**

"SplashTmiForm" - form widget containing Splash TMI

"SplashForm" - form widget containing Splash Entry area

"cikLabel2" - label widget

**4.5.24.1.2.8 Logic flow. Single pass****4.5.24.1.2.9 Data structures. None****4.5.24.1.2.10 Local data files or database. None****4.5.24.1.2.11 Limitations. None****4.5.24.2 CSU SplashSaveNRetCB**

The SplashSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Splash window and remanage the Reports window.

**4.5.24.2.1 CSU SplashSaveNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.24.2.2 CSU SplashSaveNRetCB design.**

The information identified below represents the detailed design of the SplashSaveNRetCB CSU.

**4.5.24.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.24.2.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

**4.5.24.2.2.3 Interrupts and signals. None****4.5.24.2.2.4 Algorithms. None****4.5.24.2.2.5 Error handling. None****4.5.24.2.2.6 Data conversion. None****4.5.24.2.2.7 Use of other elements. Other elements that are used by the SplashSaveNRetCB CSU:**

"SplashTmiForm" - form widget containing Splash TMI

"SplashForm" - form widget containing Splash Entry area

"cikLabel2" - label widget

"cikLabel1" - label widget

**4.5.24.2.2.8 Logic flow. Single pass****4.5.24.2.2.9 Data structures. None****4.5.24.2.2.10 Local data files or database. None****4.5.24.2.2.11 Limitations. None**



#### **4.5.24.3 CSU SplashSndRoutCB**

The SplashSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls SplashSend with priority ROUTINE.

##### **4.5.24.3.1 CSU SplashSndRoutCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.24.3.2 CSU SplashSndRoutCB design.**

The information identified below represents the detailed design of the SplashSndRoutCB CSU.

###### **4.5.24.3.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

###### **4.5.24.3.2.2 Local data elements. None**

###### **4.5.24.3.2.3 Interrupts and signals. None**

###### **4.5.24.3.2.4 Algorithms. None**

###### **4.5.24.3.2.5 Error handling. None**

###### **4.5.24.3.2.6 Data conversion. None**

###### **4.5.24.3.2.7 Use of other elements. Other elements that are used by the SplashSndRoutCB CSU:**

"PRIORITY\_ROUTINE" - define for priority message

###### **4.5.24.3.2.8 Logic flow. Single pass**

###### **4.5.24.3.2.9 Data structures. None**

###### **4.5.24.3.2.10 Local data files or database. None**

#### 4.5.24.3.2.11 Limitations. None

### 4.5.24.4 CSU SplashSend

The SplashSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldSplash to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

#### 4.5.24.4.1 CSU SplashSend requirements.

The SplashSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirments of the DMCC system.

#### 4.5.24.4.2 CSU SplashSend design.

The information identified below represents the detailed design of the SplashSend CSU.

##### 4.5.24.4.2.1 Input/output data elements.

###### INPUTS:

"prio" - priority of the message being sent

OUTPUTS: None

##### 4.5.24.4.2.2 Local data elements.

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

##### 4.5.24.4.2.3 Interrupts and signals. None

##### 4.5.24.4.2.4 Algorithms. None

##### 4.5.24.4.2.5 Error handling. None

##### 4.5.24.4.2.6 Data conversion. None

##### 4.5.24.4.2.7 Use of other elements. Other elements that are used by the SplashSend CSU:

"SplashAdd" - information on the Splash address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call

"reuseFlag" - boolean; is this a reuse and include

4.5.24.4.2.8 Logic flow. Single pass

4.5.24.4.2.9 Data structures. None

4.5.24.4.2.10 Local data files or database. None

4.5.24.4.2.11 Limitations. None

#### 4.5.24.5 CSU SplashSndUrgCB

The SplashSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls SplashSend with priority URGENT.

##### 4.5.24.5.1 CSU SplashSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.24.5.2 CSU SplashSndUrgCB design.

The information identified below represents the detailed design of the SplashSndUrgCB CSU.

###### 4.5.24.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.24.5.2.2 Local data elements. None

4.5.24.5.2.3 Interrupts and signals. None

4.5.24.5.2.4 Algorithms. None

4.5.24.5.2.5 Error handling. None

4.5.24.5.2.6 Data conversion. None

4.5.24.5.2.7 Use of other elements. Other elements that are used by the SplashSndUrgCB CSU:  
"PRIORITY\_URGENT" - define for urgent priority

4.5.24.5.2.8 Logic flow. Single pass

4.5.24.5.2.9 Data structures. None

4.5.24.5.2.10 Local data files or database. None

4.5.24.5.2.11 Limitations. None

#### 4.5.24.6 CSU ClearSplash

The ClearSplash CSU clears all the entries of the Splash report generation screen.

##### 4.5.24.6.1 CSU ClearSplash requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.24.6.2 CSU ClearSplash design.

The information identified below represents the detailed design of the ClearSplash CSU.

###### 4.5.24.6.2.1 Input/output data elements.

INPUTS:

OUTPUTS: None

###### 4.5.24.6.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null char

###### 4.5.24.6.2.3 Interrupts and signals. None

4.5.24.6.2.4 Algorithms. None

4.5.24.6.2.5 Error handling. None

4.5.24.6.2.6 Data conversion. None

4.5.24.6.2.7 Use of other elements. Other elements that are used by the ClearSplash CSU:

"SplashAdd" - info on the selection menu for the address

"EnemyType" - info on the selection menu for the enemy type

"EnemyActivity" - info on the selection menu for the enemy activity

"Direc" - info on the selection menu for the direction

"ObsInt" - info on the selection menu for the observer intentions

"SpeedText" - widget id for speed text

"NumText" - widget id for number text

"UnitText" - widget id for unit text

"cikText1" - widget id for cik1 text

"cikText2" - widget id for cik2 text

4.5.24.6.2.8 Logic flow. Single pass

4.5.24.6.2.9 Data structures. None

4.5.24.6.2.10 Local data files or database. None

4.5.24.6.2.11 Limitations. None

#### 4.5.25 Sub-Level CSC splash\_call.

The following paragraphs under this section describe the relationship of the splash\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.25.1 CSU splashad\_moreCB

The splashad\_moreCB CSU handles the callback for the splashad More pushbutton. It displays additional selectins if they exist.

**4.5.25.1.1 CSU splashad\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.25.1.2 CSU splashad\_moreCB design.**

The information identified below represents the detailed design of the splashad\_moreCB CSU.

**4.5.25.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.25.1.2.2 Local data elements. None**

**4.5.25.1.2.3 Interrupts and signals. None**

**4.5.25.1.2.4 Algorithms. None**

**4.5.25.1.2.5 Error handling. None**

**4.5.25.1.2.6 Data conversion. None**

**4.5.25.1.2.7 Use of other elements. Other elements that are used by the splashad\_moreCB CSU:**

"SplashAdd" - info on the Splash address selection menu

**4.5.25.1.2.8 Logic flow. Single pass**

**4.5.25.1.2.9 Data structures. None**

**4.5.25.1.2.10 Local data files or database. None**

**4.5.25.1.2.11 Limitations. None**

#### 4.5.25.2 CSU splashad\_rotateCB

The splashad\_rotateCB CSU handles the callback for the splashad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.25.2.1 CSU splashad\_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.25.2.2 CSU splashad\_rotateCB design.

The information identified below represents the detailed design of the splashad\_rotateCB CSU.

###### 4.5.25.2.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.25.2.2.2 Local data elements. None

4.5.25.2.2.3 Interrupts and signals. None

4.5.25.2.2.4 Algorithms. None

4.5.25.2.2.5 Error handling. None

4.5.25.2.2.6 Data conversion. None

4.5.25.2.2.7 Use of other elements. Other elements that are used by the splashad\_rotateCB CSU:

"SplashAdd" - info on Splash address selection menu

4.5.25.2.2.8 Logic flow. Single pass

4.5.25.2.2.9 Data structures. None

4.5.25.2.2.10 Local data files or database. None

**4.5.25.2.2.11 Limitations. None****4.5.25.3 CSU PutupSplash**

The PutupSplash CSU will create the Splash report screen if necessary and will also manage the screen.

**4.5.25.3.1 CSU PutupSplash requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.25.3.2 CSU PutupSplash design.**

The information identified below represents the detailed design of the PutupSplash CSU.

**4.5.25.3.2.1 Input/output data elements.**

INPUTS: None

OUTPUTS: None

**4.5.25.3.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

**4.5.25.3.2.3 Interrupts and signals. None****4.5.25.3.2.4 Algorithms. None****4.5.25.3.2.5 Error handling. None****4.5.25.3.2.6 Data conversion. None****4.5.25.3.2.7 Use of other elements. Other elements that are used by the PutupSplash CSU:**

"SplashForm" - widget id of splash form



"SplashAdd" - info on Splash address selection menu  
"addressList" - list of addresses  
"SplashMoreLabel" - widget id of More label  
"SplashMoreArrow" - widget id of More arrow  
"replyFlag" - boolean; is this a reply  
"SplashTmiForm" widget id of splash TMI form

4.5.25.3.2.8 Logic flow. Single pass

4.5.25.3.2.9 Data structures. None

4.5.25.3.2.10 Local data files or database. None

4.5.25.3.2.11 Limitations. None

#### 4.5.26 Sub-Level CSC spotTmi\_call.

The following paragraphs under this section describe the relationship of the spotTmi\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.26.1 CSU SpotClearNRetCB

The SpotClearNRetCB CSU handles the callback from the Clear and Return pushbutton. It will clear and unmanage the Spot window and remanage the Reports window.

##### 4.5.26.1.1 CSU SpotClearNRetCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.26.1.2 CSU SpotClearNRetCB design.

The information identified below represents the detailed design of the SpotClearNRetCB CSU.

##### 4.5.26.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

**4.5.26.1.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

**4.5.26.1.2.3 Interrupts and signals. None**

**4.5.26.1.2.4 Algorithms. None**

**4.5.26.1.2.5 Error handling. None**

**4.5.26.1.2.6 Data conversion. None**

**4.5.26.1.2.7 Use of other elements. Other elements that are used by the SpotClearNRetCB CSU:**

"SpotTmiForm" - form widget containing Spot TMI

"SpotForm" - form widget containing Spot Entry area

"cikLabel2" - label widget

**4.5.26.1.2.8 Logic flow. Single pass**

**4.5.26.1.2.9 Data structures. None**

**4.5.26.1.2.10 Local data files or database. None**

**4.5.26.1.2.11 Limitations. None**

**4.5.26.2 CSU SpotSaveNRetCB**

The SpotSaveNRetCB CSU handles the callback from the Save and Return pushbutton. It will unmanage the Spot window and remanage the Reports window.

**4.5.26.2.1 CSU SpotSaveNRetCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.26.2.2 CSU SpotSaveNRetCB design.**

The information identified below represents the detailed design of the SpotSaveNRetCB CSU.

**4.5.26.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.26.2.2.2 Local data elements.**

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"nullChar" null character

**4.5.26.2.2.3 Interrupts and signals. None****4.5.26.2.2.4 Algorithms. None****4.5.26.2.2.5 Error handling. None****4.5.26.2.2.6 Data conversion. None****4.5.26.2.2.7 Use of other elements. Other elements that are used by the SpotSaveNRetCB CSU:**

"SpotTmiForm" - form widget containing Spot TMI

"SpotForm" - form widget containing Spot Entry area

"cikLabel2" - label widget

"cikLabel1" - label widget

**4.5.26.2.2.8 Logic flow. Single pass****4.5.26.2.2.9 Data structures. None****4.5.26.2.2.10 Local data files or database. None****4.5.26.2.2.11 Limitations. None**

### **4.5.26.3 CSU SpotSndRoutCB**

The SpotSndRoutCB CSU handles the callback from the Send Routine pushbutton. It calls SpotSend with priority ROUTINE.

#### **4.5.26.3.1 CSU SpotSndRoutCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### **4.5.26.3.2 CSU SpotSndRoutCB design.**

The information identified below represents the detailed design of the SpotSndRoutCB CSU.

##### **4.5.26.3.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: None**

##### **4.5.26.3.2.2 Local data elements. None**

##### **4.5.26.3.2.3 Interrupts and signals. None**

##### **4.5.26.3.2.4 Algorithms. None**

##### **4.5.26.3.2.5 Error handling. None**

##### **4.5.26.3.2.6 Data conversion. None**

##### **4.5.26.3.2.7 Use of other elements. Other elements that are used by the SpotSndRoutCB CSU:**

"PRIORITY\_ROUTINE" - define for priority message

##### **4.5.26.3.2.8 Logic flow. Single pass**

##### **4.5.26.3.2.9 Data structures. None**

##### **4.5.26.3.2.10 Local data files or database. None**

#### 4.5.26.3.2.11 Limitations. None

#### 4.5.26.4 CSU SpotSend

The SpotSend CSU sends a message with a given priority. It retrieves the data from the text entry area and the annotation area and retrieves the address from the address selection menu. It then sends the data to bldSpot to be sent out. It also handles the case that the message is a reply by fixing the address to that from the message being replied to. If the message is a Reuse and Include, the included message will also be sent to the address selected.

##### 4.5.26.4.1 CSU SpotSend requirements.

The SpotSend CSU satisfies section 3.2.1.2.2.2.1, 3.2.1.2.2.3.6.1 and 3.2.1.2.2.3.6.1 of the specific requirements of the DMCC system.

##### 4.5.26.4.2 CSU SpotSend design.

The information identified below represents the detailed design of the SpotSend CSU.

##### 4.5.26.4.2.1 Input/output data elements.

###### INPUTS:

"prio" - priority of the message being sent

###### OUTPUTS: None

##### 4.5.26.4.2.2 Local data elements.

"targetCEOI" - name of user message is being sent to

"annotation" - text of annotation string

"ObserverLocation" - location of observer

"TargetQuantity" - quantity of targets

"ObserverIntentions" - intentions of observers

"TargetType" - type of target

"Target Activity" - type of activity

"locn\_string" - string containing location information

"num\_string" - string containing number information

"num" - integer containing number information

"speed\_string" - string containing speed information

"speed" - integer containing the speed

"TargetSpeed" - speed of the target

"TargetLocation" - location of the target

"TargetUnit" - pointer to the string of the target unit

##### 4.5.26.4.2.3 Interrupts and signals. None

4.5.26.4.2.4 Algorithms. None

4.5.26.4.2.5 Error handling. None

4.5.26.4.2.6 Data conversion. None

4.5.26.4.2.7 Use of other elements. Other elements that are used by the SpotSend CSU:

"SpotAdd" - information on the Spot address selection menu

"reply flag" - boolean; is this a reply

"replyCEOI" - target name for reply call

"EnemyActivity" - information on the Enemy Activity selection menu

"ObsInt" - information on the Observer Intentions selection menu

"reuseFlag" - boolean; is this a reuse and include

4.5.26.4.2.8 Logic flow. Single pass

4.5.26.4.2.9 Data structures. None

4.5.26.4.2.10 Local data files or database. None

4.5.26.4.2.11 Limitations. None

#### 4.5.26.5. CSU SpotSndUrgCB

The SpotSndUrgCB CSU handles the callback from the Send Urgent pushbutton. It calls SpotSend with priority URGENT.

##### 4.5.26.5.1 CSU SpotSndUrgCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.26.5.2 CSU SpotSndUrgCB design.

The information identified below represents the detailed design of the SpotSndUrgCB CSU.

###### 4.5.26.5.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

OUTPUTS: None

4.5.26.5.2.2 Local data elements. None

4.5.26.5.2.3 Interrupts and signals. None

4.5.26.5.2.4 Algorithms. None

4.5.26.5.2.5 Error handling. None

4.5.26.5.2.6 Data conversion. None

4.5.26.5.2.7 Use of other elements. Other elements that are used by the SpotSndUrgCB CSU:

"PRIORITY\_URGENT" - define for urgent priority

4.5.26.5.2.8 Logic flow. Single pass

4.5.26.5.2.9 Data structures. None

4.5.26.5.2.10 Local data files or database. None

4.5.26.5.2.11 Limitations. None

#### 4.5.26.6 CSU ClearSpot

The ClearSpot CSU clears all the entries of the Spot report generation screen.

##### 4.5.26.6.1 CSU ClearSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.26.6.2 CSU ClearSpot design.

The information identified below represents the detailed design of the ClearSpot CSU.

**4.5.26.6.2.1 Input/output data elements.****INPUTS:****OUTPUTS: None****4.5.26.6.2.2 Local data elements.****"argcnt" keeps track of the number of arguments set for XtSetValues****"args" the arguments set for XtSetValues****"argok" a boolean which is used to check the success of the CONVERT call****"xmstr" local to hold a XmString****"nullChar" null char****4.5.26.6.2.3 Interrupts and signals. None****4.5.26.6.2.4 Algorithms. None****4.5.26.6.2.5 Error handling. None****4.5.26.6.2.6 Data conversion. None****4.5.26.6.2.7 Use of other elements. Other elements that are used by the ClearSpot CSU:****"SpotAdd" - info on the selection menu for the address****"EnemyType" - info on the selection menu for the enemy type****"EnemyActivity" - info on the selection menu for the enemy activity****"Direc" - info on the selection menu for the direction****"ObsInt" - info on the selection menu for the observer intentions****"SpeedText" - widget id for speed text****"NumText" - widget id for number text****"UnitText" - widget id for unit text****"cikText1" - widget id for cik1 text****"cikText2" - widget id for cik2 text****4.5.26.6.2.8 Logic flow. Single pass****4.5.26.6.2.9 Data structures. None****4.5.26.6.2.10 Local data files or database. None****4.5.26.6.2.11 Limitations. None**



#### 4.5.26.7 CSU KPH\_MPHCB

The KPH\_MPHCB CSU handles the callback from the KPH/MPH pushbutton. It toggles the units the speed is used as. In addition, it changes the label after the entry area to be the currently chosen units and the label of the pushbutton to the units currently not chosen.

##### 4.5.26.7.1 CSU KPH\_MPHCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.26.7.2 CSU KPH\_MPHCB design.

The information identified below represents the detailed design of the KPH\_MPHCB CSU.

###### 4.5.26.7.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.26.7.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

###### 4.5.26.7.2.3 Interrupts and signals. None

###### 4.5.26.7.2.4 Algorithms. None

###### 4.5.26.7.2.5 Error handling. None

###### 4.5.26.7.2.6 Data conversion. None

###### 4.5.26.7.2.7 Use of other elements. Other elements that are used by the KPH\_MPHCB CSU:

"SpotTmiKPH" - widget id for KPH button

"SpeedLabel" - widget id for KPH label

4.5.26.7.2.8 Logic flow. Single pass

4.5.26.7.2.9 Data structures. None

4.5.26.7.2.10 Local data files or database. None

4.5.26.7.2.11 Limitations. None

#### 4.5.27 Sub-Level CSC spot\_call.

The following paragraphs under this section describe the relationship of the spot\_call Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.27.1 CSU enemyActivity\_moreCB

The enemyActivityad\_moreCB CSU handles the callback for the enemyActivityad More pushbutton. It displays additional selectins if they exist.

##### 4.5.27.1.1 CSU enemyActivity\_moreCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.27.1.2 CSU enemyActivity\_moreCB design.

The information identified below represents the detailed design of the enemyActivity\_moreCB CSU.

##### 4.5.27.1.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

##### 4.5.27.1.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

4.5.27.1.2.3 Interrupts and signals. None

4.5.27.1.2.4 Algorithms. None

4.5.27.1.2.5 Error handling. None

4.5.27.1.2.6 Data conversion. None

4.5.27.1.2.7 Use of other elements. Other elements that are used by the enemyActivity\_moreCB CSU:

"EnemyActivity" - info on Enemy Activity selection menu

4.5.27.1.2.8 Logic flow. Single pass

4.5.27.1.2.9 Data structures. None

4.5.27.1.2.10 Local data files or database. None

4.5.27.1.2.11 Limitations. None

#### 4.5.27.2 CSU enemyActivity\_rotateCB

The enemyActivity\_rotateCB CSU handles the callback for the enemyActivity Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.27.2.1 CSU enemyActivity\_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.27.2.2 CSU enemyActivity\_rotateCB design.

The information identified below represents the detailed design of the enemyActivity\_rotateCB CSU.

**4.5.27.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.27.2.2.2 Local data elements. None**

**4.5.27.2.2.3 Interrupts and signals. None**

**4.5.27.2.2.4 Algorithms. None**

**4.5.27.2.2.5 Error handling. None**

**4.5.27.2.2.6 Data conversion. None**

**4.5.27.2.2.7 Use of other elements. Other elements that are used by the enemyActivity\_rotateCB CSU:**

"EnemyActivity" - info on the enemy activity selection menu

**4.5.27.2.2.8 Logic flow. Single pass**

**4.5.27.2.2.9 Data structures. None**

**4.5.27.2.2.10 Local data files or database. None**

**4.5.27.2.2.11 Limitations. None**

**4.5.27.3 CSU enemyType\_rotateCB**

The enemyType\_rotateCB CSU handles the callback for the enemyType Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.27.3.1 CSU enemyType\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.27.3.2 CSU enemyType\_rotateCB design.**

The information identified below represents the detailed design of the enemyType\_rotateCB CSU.

**4.5.27.3.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .****4.5.27.3.2.2 Local data elements. None****4.5.27.3.2.3 Interrupts and signals. None****4.5.27.3.2.4 Algorithms. None****4.5.27.3.2.5 Error handling. None****4.5.27.3.2.6 Data conversion. None****4.5.27.3.2.7 Use of other elements. Other elements that are used by the enemyType\_rotateCB CSU:**

"EnemyType" - info on the Enemy Type selection menu

"spotTime" - time of last selection

**4.5.27.3.2.8 Logic flow. Single pass****4.5.27.3.2.9 Data structures. None****4.5.27.3.2.10 Local data files or database. None****4.5.27.3.2.11 Limitations. None****4.5.27.4 CSU obsInt\_rotateCB**

The obsIntad\_rotateCB CSU handles the callback for the obsIntad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.27.4.1 CSU obsInt\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.27.4.2 CSU obsInt\_rotateCB design.**

The information identified below represents the detailed design of the obsInt\_rotateCB CSU.

**4.5.27.4.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .**

**4.5.27.4.2.2 Local data elements. None**

**4.5.27.4.2.3 Interrupts and signals. None**

**4.5.27.4.2.4 Algorithms. None**

**4.5.27.4.2.5 Error handling. None**

**4.5.27.4.2.6 Data conversion. None**

**4.5.27.4.2.7 Use of other elements. Other elements that are used by the obsInt\_rotateCB CSU:**

**4.5.27.4.2.8 Logic flow. Single pass**

**4.5.27.4.2.9 Data structures. None**

**4.5.27.4.2.10 Local data files or database. None**

**4.5.27.4.2.11 Limitations. None**

#### **4.5.27.5 CSU spotad\_moreCB**

The spotad\_moreCB CSU handles the callback for the spotad More pushbutton. It displays additional selectins if they exist.

##### **4.5.27.5.1 CSU spotad\_moreCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.27.5.2 CSU spotad\_moreCB design.**

The information identified below represents the detailed design of the spotad\_moreCB CSU.

###### **4.5.27.5.2.1 Input/output data elements.**

###### **INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### **OUTPUTS: .**

4.5.27.5.2.2 Local data elements. None

4.5.27.5.2.3 Interrupts and signals. None

4.5.27.5.2.4 Algorithms. None

4.5.27.5.2.5 Error handling. None

4.5.27.5.2.6 Data conversion. None

4.5.27.5.2.7 Use of other elements. Other elements that are used by the spotad\_moreCB CSU:

"SpotAdd" - info on the Spot address selection menu

4.5.27.5.2.8 Logic flow. Single pass

4.5.27.5.2.9 Data structures. None

4.5.27.5.2.10 Local data files or database. None

**4.5.27.5.2.11 Limitations. None****4.5.27.6 CSU spotad\_rotateCB**

The spotad\_rotateCB CSU handles the callback for the spotad Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

**4.5.27.6.1 CSU spotad\_rotateCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.27.6.2 CSU spotad\_rotateCB design.**

The information identified below represents the detailed design of the spotad\_rotateCB CSU.

**4.5.27.6.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: .****4.5.27.6.2.2 Local data elements. None****4.5.27.6.2.3 Interrupts and signals. None****4.5.27.6.2.4 Algorithms. None****4.5.27.6.2.5 Error handling. None****4.5.27.6.2.6 Data conversion. None****4.5.27.6.2.7 Use of other elements. Other elements that are used by the spotad\_rotateCB CSU:**

"SpotAdd" - info on Spot address selection menu

**4.5.27.6.2.8 Logic flow. Single pass**



4.5.27.6.2.9 Data structures. None

4.5.27.6.2.10 Local data files or database. None

4.5.27.6.2.11 Limitations. None

#### 4.5.27.7 CSU direc\_rotateCB

The direc\_rotateCB CSU handles the callback for the direc Rotate pushbutton. It changes the currently selected entry to the next entry (the first entry if it is at the end already).

##### 4.5.27.7.1 CSU direc\_rotateCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.27.7.2 CSU direc\_rotateCB design.

The information identified below represents the detailed design of the direc\_rotateCB CSU.

###### 4.5.27.7.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: .

4.5.27.7.2.2 Local data elements. None

4.5.27.7.2.3 Interrupts and signals. None

4.5.27.7.2.4 Algorithms. None

4.5.27.7.2.5 Error handling. None

4.5.27.7.2.6 Data conversion. None

4.5.27.7.2.7 Use of other elements. Other elements that are used by the direc\_rotateCB CSU:

"Direc" - info on the Direction selection menu

4.5.27.7.2.8 Logic flow. Single pass

4.5.27.7.2.9 Data structures. None

4.5.27.7.2.10 Local data files or database. None

4.5.27.7.2.11 Limitations. None

#### 4.5.27.8 CSU PutupSpot

The PutupSpot CSU will create the Spot report screen if necessary and will also manage the screen.

##### 4.5.27.8.1 CSU PutupSpot requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.27.8.2 CSU PutupSpot design.

The information identified below represents the detailed design of the PutupSpot CSU.

###### 4.5.27.8.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

###### 4.5.27.8.2.2 Local data elements.

"argcnt" keeps track of the number of arguments set for XtSetValues

"args" the arguments set for XtSetValues

"argok" a boolean which is used to check the success of the CONVERT call

"xmstr" local to hold a XmString

"entries" - number of addresses

###### 4.5.27.8.2.3 Interrupts and signals. None

###### 4.5.27.8.2.4 Algorithms. None

4.5.27.8.2.5 Error handling. None

4.5.27.8.2.6 Data conversion. None

4.5.27.8.2.7 Use of other elements. Other elements that are used by the PutupSpot CSU:

"SpotForm" - widget id of spot form

"SpotAdd" - info on Spot address selection menu

"addressList" - list of addresses

"SpotMoreLabel" - widget id of More label

"SpotMoreArrow" - widget id of More arrow

"replyFlag" - boolean; is this a reply

"SpotTmiForm" widget id of spot TMI form

4.5.27.8.2.8 Logic flow. Single pass

4.5.27.8.2.9 Data structures. None

4.5.27.8.2.10 Local data files or database. None

4.5.27.8.2.11 Limitations. None

#### **4.5.28 Sub-Level CSC sysMainCB.**

The following paragraphs under this section describe the relationship of the sysMainCB Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### **4.5.28.1 CSU SysMainMsgsCB**

The SysMainMsgsCB CSU handles the callback for bringing up the Msgs screen.

##### **4.5.28.1.1 CSU SysMainMsgsCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.28.1.2 CSU SysMainMsgsCB design.**

The information identified below represents the detailed design of the SysMainMsgsCB CSU.

**4.5.28.1.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None**

**4.5.28.1.2.2 Local data elements. None****4.5.28.1.2.3 Interrupts and signals. None****4.5.28.1.2.4 Algorithms. None****4.5.28.1.2.5 Error handling. None****4.5.28.1.2.6 Data conversion. None****4.5.28.1.2.7 Use of other elements. Other elements that are used by the SysMainMsgsCB CSU:**

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

**4.5.28.1.2.8 Logic flow. Single pass.****4.5.28.1.2.9 Data structures. None****4.5.28.1.2.10 Local data files or database. None****4.5.28.1.2.11 Limitations. None****4.5.28.2 CSU SysMainRprtCB**

The SysMainRprtCB CSU handles the callback for bringing up the Rprt screen.

**4.5.28.2.1 CSU SysMainRprtCB requirements**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.28.2.2 CSU SysMainRprtCB design.**

The information identified below represents the detailed design of the SysMainRprtCB CSU.

**4.5.28.2.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.28.2.2.2 Local data elements. None****4.5.28.2.2.3 Interrupts and signals. None****4.5.28.2.2.4 Algorithms. None****4.5.28.2.2.5 Error handling. None****4.5.28.2.2.6 Data conversion. None****4.5.28.2.2.7 Use of other elements. Other elements that are used by the SysMainRprtCB CSU:**

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

**4.5.28.2.2.8 Logic flow. Single pass.****4.5.28.2.2.9 Data structures. None****4.5.28.2.2.10 Local data files or database. None****4.5.28.2.2.11 Limitations. None****4.5.28.3 CSU SysMainAddListCB**

The SysMainAddListCB CSU handles the callback for bringing up the AddList screen.

**4.5.28.3.1 CSU SysMainAddListCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.28.3.2 CSU SysMainAddListCB design.**

The information identified below represents the detailed design of the SysMainAddListCB CSU.

**4.5.28.3.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.28.3.2.2 Local data elements. None****4.5.28.3.2.3 Interrupts and signals. None****4.5.28.3.2.4 Algorithms. None****4.5.28.3.2.5 Error handling. None****4.5.28.3.2.6 Data conversion. None****4.5.28.3.2.7 Use of other elements. Other elements that are used by the SysMainAddListCB CSU:**

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

**4.5.28.3.2.8 Logic flow. Single pass.****4.5.28.3.2.9 Data structures. None****4.5.28.3.2.10 Local data files or database. None****4.5.28.3.2.11 Limitations. None**

#### 4.5.28.4 CSU SysMainGrpListCB

The SysMainGrpListCB CSU handles the callback for bringing up the GrpList screen.

##### 4.5.28.4.1 CSU SysMainGrpListCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.28.4.2 CSU SysMainGrpListCB design.

The information identified below represents the detailed design of the SysMainGrpListCB CSU.

###### 4.5.28.4.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

###### 4.5.28.4.2.2 Local data elements. None

###### 4.5.28.4.2.3 Interrupts and signals. None

###### 4.5.28.4.2.4 Algorithms. None

###### 4.5.28.4.2.5 Error handling. None

###### 4.5.28.4.2.6 Data conversion. None

###### 4.5.28.4.2.7 Use of other elements. Other elements that are used by the SysMainGrpListCB CSU:

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

###### 4.5.28.4.2.8 Logic flow. Single pass.

###### 4.5.28.4.2.9 Data structures. None

###### 4.5.28.4.2.10 Local data files or database. None

**4.5.28.4.2.11 Limitations. None****4.5.28.5 CSU SysMainLocListCB**

The SysMainLocListCB CSU handles the callback for bringing up the LocList screen.

**4.5.28.5.1 CSU SysMainLocListCB requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.28.5.2 CSU SysMainLocListCB design.**

The information identified below represents the detailed design of the SysMainLocListCB CSU.

**4.5.28.5.2.1 Input/output data elements.****INPUTS:**

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

**OUTPUTS: None****4.5.28.5.2.2 Local data elements. None****4.5.28.5.2.3 Interrupts and signals. None****4.5.28.5.2.4 Algorithms. None****4.5.28.5.2.5 Error handling. None****4.5.28.5.2.6 Data conversion. None****4.5.28.5.2.7 Use of other elements. Other elements that are used by the SysMainLocListCB CSU:**

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

**4.5.28.5.2.8 Logic flow. Single pass.**



4.5.28.5.2.9 Data structures. None

4.5.28.5.2.10 Local data files or database. None

4.5.28.5.2.11 Limitations. None

#### 4.5.28.6 CSU SysMainLogoutCB

##### 4.5.28.6.1 CSU SysMainLogoutCB requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.28.6.2 CSU SysMainLogoutCB design.

The information identified below represents the detailed design of the SysMainLogoutCB CSU.

###### 4.5.28.6.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.28.6.2.2 Local data elements. None

4.5.28.6.2.3 Interrupts and signals. None

4.5.28.6.2.4 Algorithms. None

4.5.28.6.2.5 Error handling. None

4.5.28.6.2.6 Data conversion. None

4.5.28.6.2.7 Use of other elements. Other elements that are used by the SysMainLogoutCB CSU:

None

4.5.28.6.2.8 Logic flow. Single pass.

4.5.28.6.2.9 Data structures. None

4.5.28.6.2.10 Local data files or database. None

4.5.28.6.2.11 Limitations. None

#### 4.5.28.7 CSU PutupSysMain

##### 4.5.28.7.1 CSU PutupSysMain requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.28.7.2 CSU PutupSysMain design.

The information identified below represents the detailed design of the PutupSysMain CSU.

###### 4.5.28.7.2.1 Input/output data elements.

###### INPUTS:

"w" contains widget id of button pressed

"client" contains client data

"call" contains X Windows data about the call reason

###### OUTPUTS: None

4.5.28.7.2.2 Local data elements. None

4.5.28.7.2.3 Interrupts and signals. None

4.5.28.7.2.4 Algorithms. None

4.5.28.7.2.5 Error handling. None

4.5.28.7.2.6 Data conversion. None

4.5.28.7.2.7 Use of other elements. Other elements that are used by the PutupSysMain CSU:

"sysMainTmi" - widget id of TMI area of sysMain form

"sysMainForm" - widget id of entry area of sysMain form

"tmiScreen" - widget id of current TMI

"smdScreen" - widget id of current entry screen

4.5.28.7.2.8 Logic flow. Single pass.

4.5.28.7.2.9 Data structures. None

4.5.28.7.2.10 Local data files or database. None

4.5.28.7.2.11 Limitations. None

#### 4.5.29 Sub-Level CSC util.

The following paragraphs under this section describe the relationship of the util Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC.

##### 4.5.29.1 CSU init\_selections

The init\_selections CSU initializes selection menus.

##### 4.5.29.1.1 CSU init\_selections requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.29.1.2 CSU init\_selections design.

The information identified below represents the detailed design of the init\_selections CSU.

##### 4.5.29.1.2.1 Input/output data elements.

INPUTS: None

OUTPUTS: None

##### 4.5.29.1.2.2 Local data elements. None

##### 4.5.29.1.2.3 Interrupts and signals. None

##### 4.5.29.1.2.4 Algorithms. None

4.5.29.1.2.5 Error handling. None

4.5.29.1.2.6 Data conversion. None

4.5.29.1.2.7 Use of other elements. Other elements that are used by the init\_selections CSU:

4.5.29.1.2.8 Logic flow. Single pass.

4.5.29.1.2.9 Data structures. None

4.5.29.1.2.10 Local data files or database. None

4.5.29.1.2.11 Limitations. None

#### 4.5.29.2 CSU rotate

The rotate CSU handles the rotations for all of the selection menus.

##### 4.5.29.2.1 CSU rotate requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.5.29.2.2 CSU rotate design.

The information identified below represents the detailed design of the rotate CSU.

###### 4.5.29.2.2.1 Input/output data elements.

###### INPUTS:

"panel" - panel menu to be acted upon

###### OUTPUTS: None

###### 4.5.29.2.2.2 Local data elements.

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

- 4.5.29.2.2.3 Interrupts and signals. None
- 4.5.29.2.2.4 Algorithms. None
- 4.5.29.2.2.5 Error handling. None
- 4.5.29.2.2.6 Data conversion. None
- 4.5.29.2.2.7 Use of other elements. Other elements that are used by the rotate CSU:
- 4.5.29.2.2.8 Logic flow. Single pass.
- 4.5.29.2.2.9 Data structures. None
- 4.5.29.2.2.10 Local data files or database. None
- 4.5.29.2.2.11 Limitations. None

#### **4.5.29.3 CSU moreAddress**

The moreAddress CSU handles the more button calls for the address selection menus from all the reports.

##### **4.5.29.3.1 CSU moreAddress requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.5.29.3.2 CSU moreAddress design.**

The information identified below represents the detailed design of the moreAddress CSU.

###### **4.5.29.3.2.1 Input/output data elements.**

###### **INPUTS:**

"panel" - panel menu to be acted upon

OUTPUTS: None

**4.5.29.3.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

**4.5.29.3.2.3 Interrupts and signals. None****4.5.29.3.2.4 Algorithms. None****4.5.29.3.2.5 Error handling. None****4.5.29.3.2.6 Data conversion. None****4.5.29.3.2.7 Use of other elements. Other elements that are used by the moreAddress CSU:****4.5.29.3.2.8 Logic flow. Single pass.****4.5.29.3.2.9 Data structures. None****4.5.29.3.2.10 Local data files or database. None****4.5.29.3.2.11 Limitations. None****4.5.29.4 CSU moreLctn**

The moreLctn CSU handles the more button calls for the location selection menus from all the reports.

**4.5.29.4.1 CSU moreLctn requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.29.4.2 CSU moreLctn design.**

The information identified below represents the detailed design of the moreLctn CSU.

**4.5.29.4.2.1 Input/output data elements.****INPUTS:**

"panel" - panel menu to be acted upon

**OUTPUTS:** None

**4.5.29.4.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

**4.5.29.4.2.3 Interrupts and signals. None****4.5.29.4.2.4 Algorithms. None****4.5.29.4.2.5 Error handling. None****4.5.29.4.2.6 Data conversion. None****4.5.29.4.2.7 Use of other elements. Other elements that are used by the moreLctn CSU:****4.5.29.4.2.8 Logic flow. Single pass.****4.5.29.4.2.9 Data structures. None****4.5.29.4.2.10 Local data files or database. None****4.5.29.4.2.11 Limitations. None****4.5.29.5 CSU string2utm**

The string2utm CSU converts string versions of UTM locations into the UTM structure type.

**4.5.29.5.1 CSU string2utm requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.5.29.5.2 CSU string2utm design.**

The information identified below represents the detailed design of the string2utm CSU.

**4.5.29.5.2.1 Input/output data elements.****INPUTS:**

"string" - string version of location

**OUTPUTS:**

"UTM" - UTM structure with the location

**4.5.29.5.2.2 Local data elements.**

"length" - length of the string

"index" - counter

"valid" - boolean; is this a valid UTM

**4.5.29.5.2.3 Interrupts and signals. None****4.5.29.5.2.4 Algorithms. None****4.5.29.5.2.5 Error handling. None****4.5.29.5.2.6 Data conversion. None****4.5.29.5.2.7 Use of other elements. Other elements that are used by the string2utm CSU:****4.5.29.5.2.8 Logic flow. Single pass.****4.5.29.5.2.9 Data structures. None****4.5.29.5.2.10 Local data files or database. None****4.5.29.5.2.11 Limitations. None****4.5.29.6 CSU UpdateEnvelope**

The UpdateEnvelope CSU updates both envelope icons to reflect the current number of messages that are unread.

**4.5.29.6.1 CSU UpdateEnvelope requirements.**

The UpdateEnvelope CSU satisfies section 3.2.1.2.2.1.4 of the specific requirements of the DMCC system.



**4.5.29.6.2 CSU UpdateEnvelope design.**

The information identified below represents the detailed design of the UpdateEnvelope CSU.

**4.5.29.6.2.1 Input/output data elements.**

INPUTS: None

OUTPUTS: None

**4.5.29.6.2.2 Local data elements.**

"argcnt" - keeps track of the number of arguments set for XtSetValues

"args" - the arguments set for XtSetValues

"argok" - a boolean which is used to check the success of the CONVERT call

"xmstr" - local to hold a XmString

**4.5.29.6.2.3 Interrupts and signals. None****4.5.29.6.2.4 Algorithms. None****4.5.29.6.2.5 Error handling. None****4.5.29.6.2.6 Data conversion. None****4.5.29.6.2.7 Use of other elements. Other elements that are used by the UpdateEnvelope CSU:**

"numMsgLabel" - widget id of the Msgs screen envelope

"envLabel2" - widget id of the SysMain screen envelope

**4.5.29.6.2.8 Logic flow. Single pass.****4.5.29.6.2.9 Data structures. None****4.5.29.6.2.10 Local data files or database. None****4.5.29.6.2.11 Limitations. None**

#### **4.5.30 Sub-Level CSC Builder Xcessory Functions.**

The following paragraphs under this section describe the relationship of the Builder Xcessory Functions Sub-Level CSC in terms of data flow between the CSU's of this Sub-level CSC. The functions contained herein are generated by the tool Builder Xcessory and are not modified by the user, therefore there is no information on the design of the code.

##### **4.5.30.1. CSU CreateAddrList**

The CreateAddrList CSU creates the Address List entry screen.

##### **4.5.30.2. CSU Createform requirements.**

The Createform CSU creates the toplevel form screen.

##### **4.5.30.3. CSU formTrim**

The formTrim CSU creates the decorations on most of the screens.

##### **4.5.30.4. CSU CreateFreeTxtForm**

The CreateFreeTxtForm CSU creates the Free Text entry screen.

##### **4.5.30.5. CSU CreateGrpListForm**

The CreateGrpListForm CSU creates the CEOI/Group List entry screen.

##### **4.5.30.6. CSU CreateLocListForm**

The CreateLocListForm CSU creates the Location List entry screen.

##### **4.5.30.7 CSU CreateLogonForm**

The CreateLogonForm CSU creates the Logon screen.

##### **4.5.30.8 CSU CreatemovcmdTmi**

The CreatemovcmdTmi CSU creates the TMI portion of the Movcmd report generation screen.

##### **4.5.30.9 CSU CreatemovcmdForm**

The CreatemovcmdForm CSU creates the entry portion of the Movcmd report generation screen.

##### **4.5.30.10 CSU CreateMsg1Form**

The CreateMsg1Form CSU creates the Msg1screen.

**4.5.30.11 CSU CreateMsgReadForm**

The CreateMsgReadForm CSU creates the Message Read screen.

**4.5.30.12 CSU CreatemtoTmi**

The CreatemtoTmi CSU creates the TMI portion of the MTO report generation screen.

**4.5.30.13 CSU CreatemtoForm**

The CreatemtoForm CSU creates the entry portion of the MTO report generation screen.

**4.5.30.14 CSU CreateReportForm**

The CreateReportForm CSU creates the Reports screen.

**4.5.30.15 CSU CreatereqtTmi**

The CreatereqtTmi CSU creates the TMI portion of the Request report generation screen.

**4.5.30.16 CSU CreatereqtForm**

The CreatereqtForm CSU creates the entry portion of the Request report generation screen.

**4.5.30.17 CSU CreatereuseTmi**

The CreatereuseTmi CSU creates the TMI portion of the Reuse report generation screen.

**4.5.30.18 CSU CreatereuseForm**

The CreatereuseForm CSU creates the entry portion of the Reuse report generation screen.

**4.5.30.19 CSU CreateshotTmi**

The CreateshotTmi CSU creates the TMI portion of the Shot report generation screen.

**4.5.30.20 CSU CreateshotForm**

The CreateshotForm CSU creates the entry portion of the Shot report generation screen.

#### **4.5.30.21 CSU CreatesplashTmi**

The CreatesplashTmi CSU creates the TMI portion of the Splash report generation screen.

#### **4.5.30.22 CSU CreatesplashForm**

The CreatesplashForm CSU creates the entry portion of the Splash report generation screen.

#### **4.5.30.23. CSU CreatespotTmi**

The CreatespotTmi CSU creates the TMI portion of the Spot report generation screen.

#### **4.5.30.24 CSU CreatespotForm**

The CreatespotForm CSU creates the entry portion of the Spot report generation screen.

#### **4.5.30.25 CSU CreateSysMainForm**

The CreateSysMainForm CSU creates the Sys Main screen.

### **4.6 CSC pdu decode**

The following paragraphs under this section describe the relationship of the pdu decode CSC in terms of data flow between the CSU's of this CSC and identifies all CSU interfaces that are external to the pdu decode CSC.

CSC pdu decode partially satisfies System Segment Specification requirements 3.2.1.2.1.3 and 3.2.1.2.2.1.

The following diagram depicts the structure of the pdu\_decode CSC:

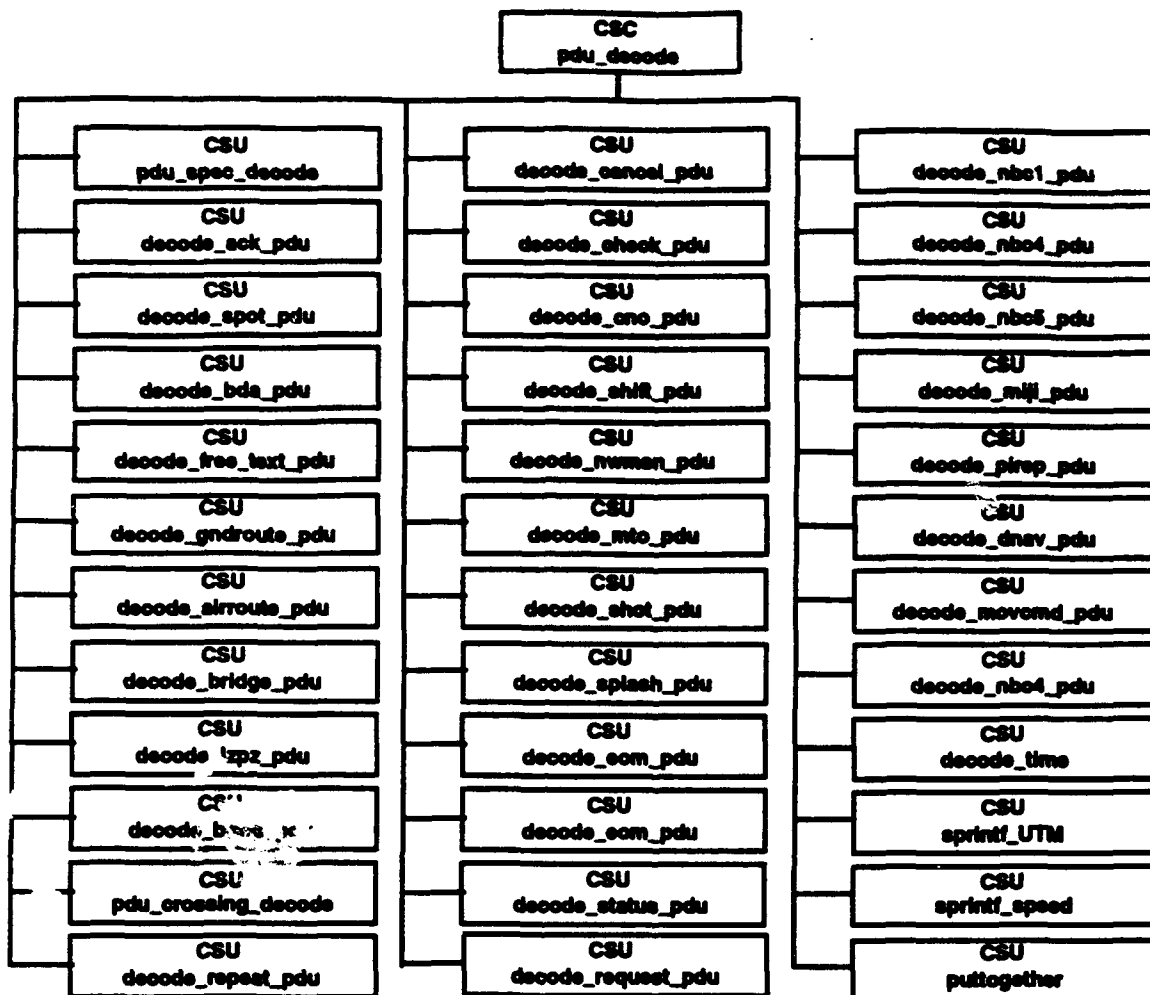


Figure 12 CSU pdu\_decode Structure

#### 4.6.1 CSU pdu\_decode.

The pdu\_decode CSU calls the appropriate function to decode the input message. The following paragraphs provide design information for this CSU.

##### 4.6.1.1 CSU pdu\_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.1.2 CSU pdu\_decode design.

The information identified below represents the detailed design of the pdu\_decode CSU.

output = pdu\_decode (input message pointer of DMCPDU\_type)

a. Input/output data elements.

INPUTS:

"pdu" is a DMCPDU\_type pointer containing the input message to be decoded.

OUTPUTS:

"dmc\_buffer" is a static character buffer of size (18+15)x36 containing the decoded message to be returned.

b. Local data elements.

"hdr\_str" is a character pointer to a buffer that contains the message header information.

"msg\_str" is a character pointer to a buffer that contains the message information.

"header\_common" is a Non\_specific pointer that points to the common header structure.

"subtype" is an integer storage that contains the pdu subtype id.

"variation" is an integer storage that contains the pdu variation id.

"good\_pdu" is a boolean flag of integer type.

c. Interrupts and signals. None

d. Algorithms.

decode and append the message variation type to ret\_str based upon its subtype.

decode and append the following fields to ret\_str in order:

Based upon the input message subtype and variation field, invoke pdu\_spec\_decode() and one of the following :

1. decode\_ack\_pdu() with an Ack\_type input message pointer.
2. decode\_spot\_pdu( ) with a Spot\_type input message pointer.
3. decode\_bda\_pdu() with an BDA\_type input message pointer
4. decode\_free\_text() wiht a FreeText\_type input message pointer
5. decode\_gndroute\_pdu() with a Recon\_type input message pointer
6. decode\_airroute\_pdu() with a Recon\_type input message pointer
7. decode\_bridge\_pdu() with a Recon\_type input message pointer
8. decode\_lzpz\_pdu() with a Recon\_type input message pointer

9. decode\_bpop\_pdu() with a Recon\_type input message pointer
10. decode\_crossing\_pdu() with a Recon\_type input message pointer
11. decode\_repeat\_pdu() with an Artillery\_type input message pointer
12. decode\_cancel\_pdu() with an Artillery\_type input message pointer
13. decode\_check\_pdu() with an Artillery\_type input message pointer
14. decode\_cno\_pdu() with an Artillery\_type input message pointer
15. decode\_shift\_pdu() with an Artillery\_type input message pointer
16. decode\_nwmsn\_pdu() an Artillery\_type input message pointer
17. decode\_mto\_pdu() with an Artillery\_type input message pointer
18. decode\_shot\_pdu() with an Artillery\_type input message pointer
19. decode\_splash\_pdu() with an Artillery\_type input message pointer
20. decode\_eom\_pdu() with an Artillery\_type input message pointer

e. Error handling. Use of system 'printf' statement to output error message to console screen.

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_spot\_label,  
arget\_enum\_str,  
activity\_enum\_str,  
direction\_enum\_str,  
obs\_int\_enum\_str.

Library routines:

switch - case,  
strcat.

CSUs:

pdu\_spec\_decode,  
decode\_ack\_pdu,  
decode\_spot\_pdu,  
decode\_bda\_pdu,  
decode\_free\_text\_pdu,  
decode\_gndroute\_pdu,  
decode\_airroute\_pdu,  
decode\_bridge\_pdu,  
decode\_lzpz\_pdu,  
decode\_bpop\_pdu,  
decode\_crossing\_pdu,  
decode\_repeat\_pdu,  
decode\_cancel\_pdu,  
decode\_check\_pdu,  
decode\_cno\_pdu,  
decode\_shift\_pdu,

decode\_nwmsn\_pdu,  
decode\_mto\_pdu,  
decode\_shot\_pdu,  
decode\_splash\_pdu,  
decode\_eom\_pdu,  
decode\_status\_pdu,  
decode\_request\_pdu,  
decode\_nbc1\_pdu,  
decode\_nbc4\_pdu,  
decode\_nbc5\_pdu,  
decode\_miji\_pdu,  
decode\_pirep\_pdu,  
decode\_dnav\_pdu,  
decode\_movcmd\_pdu.

h. Logic flow.

map header\_common template to Non\_specific input message pointer;

retrieve subtype and variation value from input message(header\_common template);

set good\_pdu to true;

Switch on input message subtype

case Acknowledgement invoke

pdu\_spec\_decode;

decode\_ack\_pdu;

case Spot invoke

pdu\_spec\_decode;

decode\_spot\_pdu;

case Battle Damage Assessment invoke

pdu\_spec\_decode;

decode\_bda\_pdu;

case Free Text invoke

pdu\_spec\_decode;

decode\_free-text\_pdu;

case Reconnaissance invoke

switch on message variation

case ground route reconnaissance report invoke

pdu\_spec\_decode;

decode\_gndroute\_pdu;

case air route reconnaissance report invoke

pdu\_spec\_decode;

decode\_airroute\_pdu;

case bridge reconnaissance report invoke

pdu\_spec\_decode;

decode\_bridge\_pdu;

case landing zone/pickup zone invoke



```
pdu_spec_decode;  
decode_lzpz_pdu;  
case BP/OP invoke  
    pdu_spec_decode;  
    decode_bpop_pdu;  
case crossing invoke  
    pdu_spec_decode;  
    decode_crossing_pdu;  
case Artillery invoke  
    switch on header_common->Common.Variation  
case repeat invoke  
    pdu_spec_decode;  
    decode_repeat_pdu;  
case cancel invoke  
    pdu_spec_decode;  
    decode_cancel_pdu;  
case check invoke  
    pdu_spec_decode;  
    decode_check_pdu;  
case cno invoke  
    pdu_spec_decode;  
    decode_cno_pdu;  
case shift invoke  
    pdu_spec_decode;  
    decode_shift_pdu;  
case new mission invoke  
    pdu_spec_decode;  
    decode_nwmsn_pdu;  
case message to observer invoke  
    pdu_spec_decode;  
    decode_mto_pdu;  
case shot invoke  
    pdu_spec_decode;  
    decode_shot_pdu;  
case splash invoke  
    pdu_spec_decode;  
    decode_splash_pdu;  
case end of mission invoke  
    pdu_spec_decode;  
    decode_eom_pdu;  
case status invoke  
    pdu_spec_decode;  
    decode_status_pdu;  
case request invoke  
    pdu_spec_decode;  
    decode_request_pdu;
```

```
case Nuclear, Biological, Chemical invoke
  switch on variation
  case Nuclear, Biological, Chemical Type 1 invoke
    pdu_spec_decode;
    decode_nbc1_pdu;
  case Nuclear, Biological, Chemical Type 4 invoke
    pdu_spec_decode;
    decode_nbc4_pdu;
  case Nuclear, Biological, Chemical Type 5 invoke
    pdu_spec_decode;
    decode_nbc5_pdu;
case Meaconing, Intrusion, Jamming, Interface
  pdu_spec_decode;
  decode_miji_pdu;
case Pilot Report invoke
  pdu_spec_decode;
  decode_pirep_pdu;
case Down Air Vehicle invoke
  pdu_spec_decode;
  decode_dnav_pdu;
case Move Comand invoke
  pdu_spec_decode;
  decode_movcmd_pdu;
return pdu_spec_decode to hdr_str;
return decode message to msg_str;
display error message for invalid subtype;
append the decoded message hdr_str and msg_str strings to
  dmc_buffer;
return dmc_buffer to caller;
```

- i. Data structures. None.
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.2 CSU pdu\_spec\_decode

The pdu\_spec\_decode CSU decodes the input message's header information. The following paragraphs provide design information for this CSU.

#### 4.6.2.1 CSU pdu\_spec\_decode requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.2.2 CSU pdu\_spec\_decode design.

The information identified below represents the detailed design of the pdu\_spec\_decode CSU.

output = pdu\_spec\_decode( input message)

a. Input/output data elements.

INPUTS:

"pdu" is a DMCPDU\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"str\_ptr" is a static character pointer that points to a temporary decoded UTM message.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"header\_common" is a Non\_specific pointer that points to the common header structure.

"temp" is a temporary storage that contains the integer value of a message field.

"subtype" is an integer storage that contains the pdu subtype id.

"variation" is an integer storage that contains the pdu variation id.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

decode and append the message variation type to ret\_str based upon its subtype.

decode and append the following fields to ret\_str in order:

message priority,

message sender CEOI,  
 message receiver,  
 message forward by,  
 message sent time/date,  
 message transmit location,  
 message transmit altitude in feet.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode_recon_variation,	decode_artillery_variation,
decode_nbc_variation,	
decode_subtype,	
decode_header_label.	

Library routines:

sprintf,  
 strcat,  
 strncat.

CSUs:

sprintf\_UTM,  
 decode\_time.

h. Logic flow.

```

set header_common template to Non_specific * pdu;
retrieve message subtype and variation value from input message;
place a null in the first byte of ret_str;
switch on subtype
case Reconnaissance pdu
  append its variation to ret_str;
case Artillery pdu
  append its variation to ret_str;
case Nuclear, Biological, Chemical type pdu
  append its variation to ret_str;
others, invoke decode_subtype[subtype] to the
  return buffer ret_str;
```

```

if message Priority is true
  append "urgency" to ret_str;
else append "routine" to ret_str;
append a carriage return to ret_str;
```

append the Sender CEOI title, the Sender CEOI string, and a carriage return to ret\_str;

append the Target CEOI title, the Target CEOI string, and a carriage return to ret\_str;

append the Last Forward CEOI title, the Last Forward CEOI string, and a carriage return to ret\_str;

invoke decode\_time() to decode the Send time;  
append the Send time title, the decoded Send time, and a carriage return to ret\_str;

invoke sprintf\_UTM() to decode the transmit location;  
append the [transmit location title, the sprintf\_UTM result, and a carriage return to ret\_str.

convert Altitude into a string with "feet" as the unit;  
append the Altitude title, the converted Altitude string, and a carriage return to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.6.3 CSU decode\_ack\_pdu**

The decode\_ack\_pdu CSU decodes the Acknowledgement input message. The following paragraphs provide design information for this CSU.

##### **4.6.3.1 CSU decode\_ack\_pdu requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.6.3.2 CSU decode\_ack\_pdu design.**

The information identified below represents the detailed design of the decode\_ack\_pdu CSU.

Syntax for invoking this function:

output = decode\_ack\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a Ack\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a static character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Acknowledge CEOI;

Original DTG;

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - constant char:

decode\_ack\_label - two dimensional array of size 15x16,

Library routines:

sprintf,

strcat,

strncat.

CSUs:

decode\_time.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

append the Acknowledge CEOI title, the Acknowledge CEOI string, and a carriage return to ret\_str;

invoke decode\_time to decode the Original DTG;

append the Original DTG title, the decode Original DTG string, and a carriage return to ret\_str;

append nine carriage returns to ret\_str( nine blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.4 CSU decode\_spot\_pdu

The decode\_spot\_pdu CSU decodes the Spot input message. The following paragraphs provide design information for this CSU.

##### 4.6.4.1 CSU decode\_spot\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.4.2 CSU decode\_spot\_pdu design.

The information identified below represents the detailed design of the decode\_spot\_pdu CSU.

Syntax for invoking this function:

output = decode\_spot\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a Spot\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"str\_ptr" is a static character pointer that points to the temporary decoded UTM message.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Target Quantity,  
Target Type,  
Target Activity Type,  
Target Speed,  
Target Unit,  
Target Location,  
Target Sighted Time,  
Observer Location,  
Observer Intention,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_spot\_label,  
arget\_enum\_str,



activity\_enum\_str,  
direction\_enum\_str,  
obs\_int\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether,  
sprintf\_speed,  
sprintf\_UTM,  
decode\_time.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

convert Target Quantity into a string;

append the Target Quantity title and the Target Quantity string to  
ret\_str with a carriage return at the end;

if TargetType is greater than or equal to zero and it is less than the  
number of possible Target Type.

decode Target Type;

else

decode Target Type with an invalid\_entry string;

invoke puttogether() to append the Target Type title, the decoded  
Target Type string and a carriage return to ret\_str;

if Target Activity is greater than or equal to zero and it is less than the  
number of possible Target Activity Type.

decode Target Activity;

else

decode Target Type with an invalid\_entry string;

invoke puttogether() to append the Target Activity title, the decoded  
Target Activity string and a carriage return to ret\_str;

invoke sprintf\_speed() to decode the Target Speed;

append the Target Speed title, the decoded Target Speed string and a  
carriage return to ret\_str;

if Target Direction is greater than or equal to zero and it is less than the  
number of possible Target Direction.

decode Target Direction;

else

decode Target Direction with an invalid\_entry string;

invoke puttogether() to append the Target Direction title, the decoded Target Direction string and a carriage return to ret\_str;

append the Target Unit title, the decode Target Unit and a carriage return to ret\_str;

invoke sprintf\_UTM() to decode Target Location;  
append the Target Location title, the decoded Target Location string and a carriage return to ret\_str;

invoke decode\_time() to decode Time Sighted;  
append the Target Time Sighted title, the decoded Time Sighted string and a carriage return to ret\_str;

invoke sprintf\_UTM() to decode Observer Location;  
append the Observer Location title, the decoded Observer Location string and a carriage return to ret\_str;

if Observer Intention is greater than or equal to zero and it is less than the number of possible Observer Intentions.

    decode Observer Intention;

else

    decode Observer Intention with an invalid\_entry string;

invoke puttogether() to append the Observer Intention title, the decoded Observer Intention string and a carriage return to ret\_str;

append a carriage return for a blank line;

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str.

- i.     Data structures. None
- j.     Local data files or database. None
- k.     Limitations. None

#### 4.6.5 CSU decode\_bda\_pdu

The decode\_bda\_pdu CSU decodes the Battle Damage Assessment input message. The following paragraphs provide design information for this CSU.

#### 4.6.5.1 CSU decode\_bda\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.5.2 CSU decode\_bda\_pdu design.

The information identified below represents the detailed design of the decode\_bda\_pdu CSU.

Syntax for invoking this function:

output = decode\_bda\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a Battle Damage Assessment (BDA\_type) type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a static character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporarily decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Strike Start Time,

Strike End Time,

Target Category,

Target Type,

# of Target Destroyed,

# of Percent Coverage,

Free Text Annotation String.

\*append a carriage return at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types -

character pointer:

decode\_bda\_str,

target\_enum\_str,

percent\_cover\_enum\_str.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

puttogether,

decode\_time.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

invoke decode\_time() to decode the start of Strike Time;

append the Strike Time Start title, the decode time and a carriage return to ret\_str;

invoke decode\_time() to decode the end of Strike Time;

append the Strike Time End title, the decode time and a carriage return to ret\_str;

append the Target Category title and a carriage return to ret\_str.

if TargetType is greater than or equal to zero and it is less than the number of possible Target Type.

decode Target Type;

else

decode Target Type with an invalid\_entry string;

invoke puttogether() to append the Target Type title, the decoded Target Type string and a carriage return to ret\_str;

convert the Target Destroyed into a string;

append the Target Destroyed title, the Target Destroyed string and a carriage return to ret\_str;

if Percent Coverage type is greater than or equal to zero and it is less than the number of possible Percent Coverage type.

decode Percent Coverage type;

else

decode Percent Coverate with an invalid\_entry string;

invoke puttogether() to append the Percent Coverage title, the decoded Percent Coverage string and a carriage return to ret\_str;

append five carriage return to ret\_str(for five blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.6 CSU decode\_free\_text\_pdu

The decode\_free\_text\_pdu CSU decodes the Free Text input message. The following paragraphs provide design information for this CSU.

##### 4.6.6.1 CSU decode\_free\_text\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.6.2 CSU decode\_free\_text\_pdu design.

The information identified below represents the detailed design of the decode\_free\_text\_pdu CSU.

Syntax for invoking this function:

output = decode\_free\_text\_pdu( pdu);

a. Input/output data elements.

##### INPUTS:

" pdu" is a Free Text (FreeText\_type) type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a static character buffer of size 18x36 containing the decoded message string to be returned.

- b. Local data elements. None
- c. Interrupts and signals. None
- d. Algorithms.  
store a null in the first byte of ret\_str.  
decode and append the Free Text string to ret\_str;
- e. Error handling. None
- f. Data conversion. None
- g. Use of other elements.  
Data elements: None  
Library routines:  
    sprintf,  
    strcat,  
    strncat.  
CSUs: None
- h. Logic flow.  
set ret\_str to null;  
append the Free Text string and a carriage return to ret\_str;  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

**4.6.7 CSU decode\_gndroute\_pdu**

The decode\_gndroute\_pdu CSU decodes the Ground Route input message. The following paragraphs provide design information for this CSU.

#### 4.6.7.1 CSU decode\_gndroute\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.7.2 CSU decode\_gndroute\_pdu design.

The information identified below represents the detailed design of the decode\_gndroute\_pdu CSU.

Syntax for invoking this function:

output = decode\_bda\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a ground route reconaissance(Recon \_type) type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a static character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

"temp\_char" is a temporary unsigned character storage.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Enemy Activity,

Classification Formula,

Classification Formula Information,

Route ID,

Free Text Annotation String.

\*append a carriage return at the end of each field.

- e. Error handling. None
- f. Data conversion. None
- g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_gndroute,  
classif\_enum\_str,  
activity\_enum\_str,

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether().

- h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

if Enemy Activity is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.

decode Enemy Activity Type;

else

decode Enemy Activity Type with an invalid\_entry string;

invoke puttogether() to append the Enemy Activity title, the decoded Enemy Activity Type string and a carriage return to ret\_str;

if Classification Formula is greater than or equal to zero and it is less than the number of possible Classification Formula Type.

decode Classification Formula Type;

else

decode Classification Formula Type with an invalid\_entry string;

invoke puttogether() to append the Classification Formula title, the decoded Classification Formula Type string and a carriage return to ret\_str;

append the Classification Information title, the Classification Formular Information string, and a carriage return to ret\_str;

append the Route ID title, the Route ID string, and a carriage return to ret\_str;

append seven carriage return to ret\_str(seven blank lines);  
concatenate the first line(32 characters) of the



pdu->Common.FreeText and a carriage into ret\_str;  
concatenate the next line(32 characters) of the  
pdu->Common.FreeText + 32 and a carriage into ret\_str;

return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.8 CSU decode\_airroute\_pdu

The decode\_airroute\_pdu CSU decodes the Air Route input message. The following paragraphs provide design information for this CSU.

##### 4.6.8.1 CSU decode\_airroute\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.8.2 CSU decode\_airroute\_pdu design.

The information identified below represents the detailed design of the decode\_airroute\_pdu CSU.

Syntax for invoking this function:

output = decode\_airroute\_pdu( pdu);

- a. Input/output data elements.

**INPUTS:**

"pdu" is a airroute\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

- b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

place a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Enemy Activity,

Obstacles,

Route\_ID,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_airroute,

activity\_enum\_str,

obstacle\_enum\_str.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(one blank line);

if Enemy Activity is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.

decode Enemy Activity Type;

else

decode Enemy Activity Type with an invalid\_entry string;

invoke puttogether() to append the Enemy Activity title, the decoded Enemy Activity Type string and a carriage return to ret\_str;

if Obstacles is greater than or equal to zero and it is less than the number of possible Obstacle Type.

    decode Obstacle Type;

else

    decode Obstacle Type with an invalid\_entry string;

    invoke puttogether() to append the Obstacle title, the decoded Obstacle Type string and a carriage return to ret\_str;

append the Route ID title, the Route ID string to ret\_str, and a carriage return to ret\_str;

append six carriage return to ret\_str( for six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i.     Data structures. None

j.     Local data files or database. None

k.     Limitations. None

#### 4.6.9 CSU decode\_bridge\_pdu

The decode\_bridge\_pdu CSU decodes the Bridge Reconnaissance input message. The following paragraphs provide design information for this CSU.

##### 4.6.9.1 CSU decode\_bridge\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.9.2 CSU decode\_bridge\_pdu design.

The information identified below represents the detailed design of the decode\_bridge\_pdu CSU.

Syntax for invoking this function:

output = decode\_bridge\_pdu( pdu);

a.     Input/output data elements.

**INPUTS:**

"pdu" is a Recon\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

**b. Local data elements.**

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

**c. Interrupts and signals. None****d. Algorithms.**

store a null in the first byte of ret\_str.

decode and append the following fields to ret\_str in order:

Bridge Type,  
Damage,  
Spans,  
Construction Material,  
Bridge Length,  
Bridge Width,  
Bridge Height,  
Bridge Under,  
Bridge Construction Description,  
Bridge ID,  
Bridge Span Length,  
Bridge Load Class,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

**e. Error handling. None****f. Data conversion. None****g. Use of other elements.**

The following provide the string values of "Enumerated" types - character pointer:

decode\_bridge,  
bridge\_enum\_str,  
bridge\_damage\_enum\_str,  
bridge\_material\_enum\_str,

## Library routines:

sprintf,  
strcat,  
strncat.

## CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

if Bridge Type is greater than or equal to zero and it is less than the number of possible Bridge Type.

decode Bridge Type;

else

decode Bridge Type with an invalid\_entry string;

invoke puttogether() to append the "BRIDGE TYPE" title, the decoded Bridge Type string and a carriage return to ret\_str;

if Bridge Damage Type is greater than or equal to zero and it is less than the number of possible Bridge Damage Type.

decode Bridge Damage Type;

else

decode Bridge Damage Type with an invalid\_entry string;

invoke puttogether() to append the Damage title, the decoded Bridge Damage Type string and a carriage return to ret\_str;

convert the Bridge Spans into a string;

append the Spans title, the Bridge Spans string, and a carriage return to ret\_str;

if Bridge Construction Material Type is greater than or equal to zero and it is less than the number of possible Bridge Construction Material Type.

decode Bridge Construction Material Type;

else

decode Bridge Construction Material Type with an invalid\_entry string;

invoke puttogether() to append the Construction Material title, the decoded Bridge Construction Material Type string and a carriage return to ret\_str;

append the Length title, the Bridge Length string, and a carriage return into ret\_str;

append the Width title, the Bridge Width string and a carriage return into ret\_str;

append the Height title, the Bridge Height string and a carriage return into ret\_str;

append the Under title, the Bridge Under string and a carriage return into ret\_str;

append the Construction Description title, the Bridge Construction Description string and a carriage return into ret\_str;

append the Bridge ID title, the Bridge ID string and a carriage return into ret\_str;

append the Span Length title, the Bridge Span Length string and a carriage return into ret\_str;

append the Load Class title, the Bridge Load Class string and a carriage return into ret\_str;

append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.6.10 CSU decode\_lzpz\_pdu**

The decode\_lzpz\_pdu CSU decodes the Landing/Pickup Zone Reconnaissance input message. The following paragraphs provide design information for this CSU.

##### **4.6.10.1 CSU decode\_lzpz\_pdu requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.6.10.2 CSU decode\_lzpz\_pdu design.**

The information identified below represents the detailed design of the decode\_lzpz\_pdu CSU.

Syntax for invoking this function:

output = decode\_lzpz\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a Recon\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Activity Likely Type,  
Obstacles Type,  
Obstacles Description,  
Landing/Pickup Zone ID,  
Landing/Pickup Zone Size,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_lzpz,  
activity\_enum\_str,  
obstacle\_enum\_str,

Library routines:

sprintf,  
strcat,  
strncat.  
CSUs:  
puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

if Activity Likely Type is greater than or equal to zero and it is less than  
the number of possible Activity Likely Type.

decode Activity Likely Type;

else

decode Activity Likely Type with an invalid\_entry string;

invoke puttogether() to append the Activity Likely title, the decoded  
Activity Likely Type string and a carriage return to ret\_str;

if Obstacle is greater than or equal to zero and is less than the number  
of possible ObstacleType.

convert Obstacle into a string;

else

decode Obstacle Type with an invalid\_entry string;

invoke puttogether() to append the Obstacles title, the decoded Obstacle  
Type string and a carriage return to ret\_str;

append the Obstacle Description title, the Obstacle Description string,  
and a carriage return to ret\_str;

append the LZ/PZ ID title, the Loading/Pickup Zone ID string, and a  
carriage return to ret\_str;

append the LZ/PZ Size title, the Loading/Pickup Zone Size string, and  
a carriage return to ret\_str;

append six carriage returns to ret\_str( six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None



#### 4.6.11 CSU decode\_bpop\_pdu

The decode\_bpop\_pdu CSU decodes the bpop Reconnaissance input message. The following paragraphs provide design information for this CSU.

##### 4.6.11.1 CSU decode\_bpop\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.11.2 CSU decode\_bpop\_pdu design.

The information identified below represents the detailed design of the decode\_bpop\_pdu CSU.

Syntax for invoking this function:

```
output = decode_bpop_pdu( pdu);
```

a. Input/output data elements.

INPUTS:

" pdu" is a Recon\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Enemy Activity Type,  
Obstacles Type,  
Obstacles Description,  
BP/OP ID,

BP/OP Size,  
BP/OP Axis,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_bpop,  
activity\_enum\_str,  
obstacle\_enum\_str,

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

if Enemy Activity Type is greater than or equal to zero and it is less than the number of possible Enemy Activity Type.

decode Enemy Activity Type;

else

decode Enemy Activity Type with an invalid\_entry string;

invoke puttogether() to append the Activity Likely title, the decoded Enemy Activity Type string and a carriage return to ret\_str;

if Obstacle is greater than or equal to zero and is less than the number of possible ObstacleType.

convert Obstacle into a string;

else

decode Obstacle Type with an invalid\_entry string;

invoke puttogether() to append the Obstacle title, the decoded Obstacle Type string and a carriage return to ret\_str;

append the Obstacle Description title, the Obstacle Description string, and a carriage return to ret\_str;

append the BP/OP ID title, the BP/OP ID string, and a carriage return to ret\_str;

append the BP/OP Size title, the BP/OP Size string, and a carriage return to ret\_str;

append the Axis title, the BP/OP Axis string, and a carriage return to ret\_str;

append five carriage returns to ret\_str( five blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.12 CSU decode\_crossing\_pdu

The decode\_crossing\_pdu CSU decodes the Crossing Reconnaissance input messag.. The following paragraphs provide design information for this CSU.

##### 4.6.12.1 CSU decode\_crossing\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.12.2 CSU decode\_crossing\_pdu design.

The information identified below represents the detailed design of the decode\_crossing\_pdu CSU.

Syntax for invoking this function:

output = decode\_crossing\_pdu( pdu);

- a. Input/output data elements.

##### INPUTS:

" pdu" is a Recon\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Bank Slop Entry,  
Bank Slop Exit,  
Crossing Length,  
Crossing Width,  
Crossing Depth,  
Current Flow,  
Crossing ID,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_crossing.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

None

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Bank Slope Entry title, the Bank Slope Entry string, and a carriage return to ret\_str;

append the Bank Slope Exit title, the Bank Slope Exit string, and a carriage return to ret\_str;

append the Crossing Length title, the Crossing Length string, and a carriage return to ret\_str;

append the Crossing Width title, the Crossing Width string, and a carriage return to ret\_str;

append the Crossing Depth title, the Crossing Depth string, and a carriage return to ret\_str;

append the Current Flow title, the Current Flow string, and a carriage return to ret\_str;

append the Current ID title, the Crossing ID string, and a carriage return to ret\_str;

append four carriage returns to ret\_str( four blank lines);  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### **4.6.14 CSU decode\_repeat\_pdu**

The decode\_repeat\_pdu CSU decodes the Repeat Artillery input message. The following paragraphs provide design information for this CSU.

##### **4.6.14.1 CSU decode\_repeat\_pdu requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### **4.6.14.2 CSU decode\_repeat\_pdu design.**

The information identified below represents the detailed design of the decode\_repeat\_pdu CSU.

Syntax for invoking this function:  
output = decode\_repeat\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append five carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

repeat message,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_repeat.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

None.

h. Logic flow.

set ret\_str to null;

append five carriage returns to ret\_str;

append the "REPEAT" message string at the center of a line and a carriage return to ret\_str;

append six carriage returns to ret\_str( six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.15 CSU decode\_pirep\_pdu

The decode\_pirep\_pdu CSU decodes the Pilot Report input message. The following paragraphs provide design information for this CSU.

##### 4.6.15.1 CSU decode\_pirep\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.15.2 CSU decode\_pirep\_pdu design.

The information identified below represents the detailed design of the decode\_pirep\_pdu CSU.

Syntax for invoking this function:

output = decode\_pirep\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a PIREP\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Visibility,  
Restrictions,  
Cloud Cover,  
Cloud Base,  
Cloud Top,  
Outside Air Temperature,  
Barometer,  
Wind Speed,  
Wind Direction,  
Turbulence Frequency,  
Turbulence Intensity,  
Icing Intensity,  
Icing Type,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_pirep,  
visibility\_enum\_str,  
restriction\_enum\_str,  
cloud\_enum\_str,  
direction\_enum\_str,  
turbfreq\_enum\_str,



turbint\_enum\_str,  
iceint\_enum\_str,  
icing\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

None.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

if Visibility Type is greater than or equal to zero and is less than the  
number of possible Visibility Type.

decode Visibility Type;

else

decode Visibility Type with an invalid\_entry string;

append the Visibility title, the decoded Visibiltity Type, and a space to  
ret\_str;

if Restriction Type is greater than or equal to zero and is less than the  
number of possible RestrictionType.

decode Restriction Type into a string;

else

decode Restriction Type with an invalid\_entry string;

append the decoded RestrictionType string and a carriage return to  
ret\_str;

if Cloud Cover Type is greater than or equal to zero and is less than the  
number of possible Cloud Cover Type.

decode Cloud Cover Type into a string;

else

decode Cloud Cover Type with an invalid\_entry string;

appends the Clouds title, the decoded Cloud Cover Type string and a  
comma with a space to ret\_str;

convert Cloud Base into a string;

append the Cloud Base title, the Cloud Base string, and a comma with a  
space to ret\_str;

convert Cloud Top into a string;

append the Cloud Top title, the Cloud Top string, and two carriage  
returns to ret\_str;

convert Outside Air Temperature into a string with unit "C";  
append the Outside Air Temperature title and the Outside Air  
Temperature string to ret\_str;

convert Barometer into a string with two spaces after the decimal and  
unit """;  
append the Barometer title, the Barometer string, and two carriage  
returns to ret\_str;

convert Wind Speed into a string with unit "KNOTS";  
append the Wind title, the converted Wind Speed string, and a comma  
with a space to ret\_str;

if Wind Direction is greater than or equal to zero and is less than the  
number of possible Wind Speed.

decode Wind Speed into a string;

else

decode Wind Speed with an invalid\_entry string;

append the decoded Wind Speed string and two carriage returns to  
ret\_str;

if Turbulence Frequency is greater than or equal to zero and is less than  
the number of possible Turbulence Frequency.

decode Turbulence Frequency into a string;

else

decode Turbulence Frequency with an invalid\_entry string;

append the decoded Turbulence Frequency string and a comma to  
ret\_str;

if Turbulence Intensity Type is greater than or equal to zero and is less  
than the number of possible Turbulence Intensity Type.

decode Turbulence Intensity Type into a string;

else

decode Turbulence Intensity Type with an invalid\_entry string;

append the decoded Turbulence Intensity Type string, the Turbulence  
title, and two carriage returns to ret\_str;

if Icing Intensity is greater than or equal to zero and is less than the  
number of possible Icing Intensity.

decode Icing Intensity into a string;

else

decode Icing Intensity with an invalid\_entry string;

append the decoded Icing Intensity string and a space to ret\_str;

if Icing Type is greater than or equal to zero and is less than the number  
of possible Icing Type.

decode Icing Type into a string;  
else  
decode Icing Type with an invalid\_entry string;  
append the decoded IcingType string, the Icing title, and two carriage  
returns to ret\_str;

append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.16 CSU decode\_cancel\_pdu

The decode\_cancel\_pdu CSU decodes the Cancel Artillery input message.  
The following paragraphs provide design information for this CSU.

##### 4.6.16.1 CSU decode\_cancel\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.16.2 CSU decode\_cancel\_pdu design.

The information identified below represents the detailed design of the decode\_cancel\_pdu CSU.

Syntax for invoking this function:  
output = decode\_cancel\_pdu( pdu);

- a. Input/output data elements.

**INPUTS:**

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append five carriage returns to ret\_str.

decode and append the following fields to ret\_str in order:

cancel message string;

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types -  
character pointer:

decode\_cancel.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

None.

h. Logic flow.

set ret\_str to null;

append five carriage return to ret\_str(five blank lines);

append "CANCEL" message string to the center of a line and a carriage  
return to ret\_str;

append six carriage returns to ret\_str( six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.17 CSU decode\_check\_pdu

The decode\_check\_pdu CSU decodes the Check Artillery input message. The following paragraphs provide design information for this CSU.

##### 4.6.17.1 CSU decode\_check\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.17.2 CSU decode\_check\_pdu design.

The information identified below represents the detailed design of the decode\_check\_pdu CSU.

Syntax for invoking this function:

```
output = decode_check_pdu( pdu);
```

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size18x36 containing the decoded message string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append five carriage returns to ret\_str.

decode and append the following fields to ret\_str in order:

check message string,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:  
decode\_check.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

None.

h. Logic flow.

set ret\_str to null;

append five carriage returns to ret\_str(five blank lines);

append "CHECK" message string at the center of a line and a carriage return to ret\_str;

append six carriage returns to ret\_str( six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.18 CSU decode\_cno\_pdu

The decode\_cno\_pdu CSU decodes the Can Not Observe Artillery input message. The following paragraphs provide design information for this CSU.

##### 4.6.18.1 CSU decode\_cno\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.18.2 CSU decode\_cno\_pdu design.

The information identified below represents the detailed design of the decode\_cno\_pdu CSU.

Syntax for invoking this function:

```
output = decode_cno_pdu( pdu);
```

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Mission ID,

Target ID,

Mission Status,

Can Not Observe Message String,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_cno,

mission\_status\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

append three carriage returns to ret\_str(three blank lines);

append the "CAN NOT OBSERVE" message string to ret\_str;

append four carriage returns to ret\_str(four blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.19 CSU decode\_shift\_pdu

The decode\_shift\_pdu CSU decodes the Shift Artillery input message. The following paragraphs provide design information for this CSU.



#### 4.6.19.1 CSU decode\_shift\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.19.2 CSU decode\_shift\_pdu design.

The information identified below represents the detailed design of the decode\_shift\_pdu CSU.

Syntax for invoking this function:

output = decode\_shift\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Mission ID,

Target ID,

Mission Status,

Fire for Effect logical Type,

Shift Message String,

Shift Instruction Message String,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_shift,  
mission\_status\_enum\_str,  
logic\_enum\_str,

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

if Fire for Effect Type is greater than or equal to zero and is less than the number of possible Fire for Effect Type.

decode Fire for Effect Type into a string;

else

decode Fire for Effect Type with an invalid\_entry string;

invoke puttogether() to append the Fire For Effect title, the decoded Fire for Effect Type string and a carriage return to ret\_str;

append two carriage returns to ret\_str;

append the "SHIFT" message string at the center of a line and a carriage return to ret\_str;

append nine character spaces to ret\_str;  
append the the Shift Instruction Message string and a carriage return to ret\_str;

append two carriage returns to ret\_str( two blank lines);  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.20 CSU decode\_nwmsn\_pdu

The decode\_nwmsn\_pdu CSU decodes the New Artillery input message. The following paragraphs provide design information for this CSU.

##### 4.6.20.1 CSU decode\_nwmsn\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.20.2 CSU decode\_nwmsn\_pdu design.

The information identified below represents the detailed design of the decode\_nwmsn\_pdu CSU.

Syntax for invoking this function:

output = decode\_nwmsn\_pdu( pdu);

- a. Input/output data elements.

#### INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

New Artillery Mission title String,  
Mission ID,  
Target ID,  
Mission Status,  
Mission Type,  
Shell Type,  
Control Type,  
Fuze Type,  
Trajectory Type,  
Fire for Effect logical Type,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_nwmsn,  
mission\_status\_enum\_str,  
mission\_enum\_str,  
shell\_enum\_str,  
control\_enum\_str,  
fuze\_enum\_str,  
trajectory\_enum\_str,  
logic\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.  
CSUs:  
puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

append the " \* NEW ARTILLERY MISSION \* " message string and a carriage return to ret\_str;

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

if Mission Type is greater than or equal to zero and it is less than the number of possible Mission Type.

decode Mission Type;

else

decode Mission Type with an invalid\_entry string;

invoke puttogether() to append the Mission Type title, the decoded Mission Type string and a carriage return to ret\_str;

if Shell Type is greater than or equal to zero and it is less than the number of possible Shell Type.

decode Shell Type;

else

decode Shell Type with an invalid\_entry string;

invoke puttogether() to append the Shell Type title, the decoded Shell Type string and a carriage return to ret\_str;

if Control Type is greater than or equal to zero and it is less than the number of possible Control Type.

decode Control Type;

```
else
    decode Control Type with an invalid_entry string;
    invoke puttogether() to append the Control Type title, the decoded
    Control Type string and a carriage return to ret_str;

if Fuze Type is greater than or equal to zero and it is less than the
    number of possible Fuze Type.
    decode Fuze Type;
else
    decode Fuze Type with an invalid_entry string;
    invoke puttogether() to append the Fuze Type title, the decoded Fuze
    Type string and a carriage return to ret_str;

if Trajectory Type is greater than or equal to zero and it is less than the
    number of possible Trajectory Type.
    decode Trajectory Type;
else
    decode Trajectory Type with an invalid_entry string;
    invoke puttogether() to append the Trajectory Type title, the decoded
    Trajectory Type string and a carriage return to ret_str;

if Fire for Effect Type is greater than or equal to zero and it is less than
    the number of possible Fire for Effect Type.
    decode Fire for Effect Type;
else
    decode Fire for Effect Type with an invalid_entry string;
    invoke puttogether() to append the Fire For Effect title, the decoded
    Fire for Effect Type string and a carriage return to ret_str;

append the first 32 Free Text characters and a carriage to ret_str;
append the next 32 Free Text characters) and a carriage to ret_str;
return ret_str to caller.
```

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.21 CSU decode\_mto\_pdu

The decode\_mto\_pdu CSU decodes the Message to Observe input message. The following paragraphs provide design information for this CSU.

**4.6.21.1 CSU decode\_mto\_pdu requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.6.21.2 CSU decode\_mto\_pdu design.**

The information identified below represents the detailed design of the decode\_mto\_pdu CSU.

Syntax for invoking this function:  
output = decode\_mto\_pdu( pdu);

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Mission ID,

Target ID,

Mission Status,

Message To Observe message String,

Request Adjustment,

Enter As Target Logic Type,

End Mission Logic Type,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_mto,  
mission\_status\_enum\_str,  
logic\_enum\_str,

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

append the "MESSAGE TO OBSERVER" string at the center of a line and a carriage return to ret\_str;

if Request Adjustment Type is greater than or equal to zero and it is less than the number of possible Logical Type.

decode Request Adjustment Logical Type;

else



```
    decode Request Adjustment Logical Type with an invalid_entry
    string;
    invoke puttogether() to append the Request Adjustment title, the
    decoded Request Adjustment Logical Type string and a carriage
    return to ret_str;

    if Enter As Target Logical Type is greater than or equal to zero and it is
    less than the number of possible Logical Type.
        decode Enter As Target Logical Type;
    else
        decode Enter As Target Logical Type with an invalid_entry string;
        invoke puttogether() to append the Enter As Target title, the decoded
        Enter As Target Logical Type string and a carriage return to
        ret_str;

    if End Mission Logical Type is greater than or equal to zero and it is less
    than the number of possible Logical Type.
        decode End Mission Logical Type;
    else
        decode End Mission Logical Type with an invalid_entry string;
        invoke puttogether() to append the End Mission title , the decoded End
        Mission Logical Type string and a carriage return to ret_str;

    append two carriage returns to ret_str( two blank lines);
    append the first 32 Free Text characters and a carriage to ret_str;
    append the next 32 Free Text characters) and a carriage to ret_str;
    return ret_str to caller.
```

- i.     Data structures. None
- j.     Local data files or database. None
- k.     Limitations. None

#### 4.6.22 CSU decode\_shot\_pdu

The decode\_shot\_pdu CSU decodes the Shot Artillery input message. The following paragraphs provide design information for this CSU.

#### 4.6.22.1 CSU decode\_shot\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

#### 4.6.22.2 CSU decode\_shot\_pdu design.

The information identified below represents the detailed design of the decode\_shot\_pdu CSU.

Syntax for invoking this function:

```
output = decode_shot_pdu( pdu);
```

a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

"i" is a loop counter.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Shot Message String,

Mission ID,

Target ID,

Mission Status Type,

Shot Fire Message String,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_shot,  
mission\_status\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

append three carriage returns to ret\_str;

append the " \* SHOT FIRED \* " message at the center of a line to ret\_str;

append four carriage returns to ret\_str( four blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

- j. Local data files or database. None
- k. Limitations. None

#### 4.6.23 CSU decode\_splash\_pdu

The decode\_splash\_pdu CSU decodes the Splash Artillery input message. The following paragraphs provide design information for this CSU.

##### 4.6.23.1 CSU decode\_splash\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.23.2 CSU decode\_splash\_pdu design.

The information identified below represents the detailed design of the decode\_splash\_pdu CSU.

Syntax for invoking this function:  
output = decode\_splash\_pdu( pdu);

- a. Input/output data elements.

INPUTS:

" pdu" is a Artillery\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

- b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

- c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Mission ID,

Target ID,

Mission Status Type,

Splash message string,

Round Fired,

Second to Impact time,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_splash,

mission\_status\_enum\_str.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

append two carriage returns to ret\_str;  
append the "\*\* \* SPLASH \* \*" message at the center of a line and a carriage return to ret\_str;

convert Rounds Fired into a string;  
invoke puttogether() to append the Rounds title, the converted Rounds Fired string, and a carriage return to ret\_str;

convert Impact Time into a string;  
invoke puttogether() to append the Second to Impact title, the converted Impact Time string, and a carriage return to ret\_str;

append two carriage returns to ret\_str( two blank lines);  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.24 CSU decode\_eom\_pdu

The decode\_eom\_pdu CSU decodes the End Of Mission Artillery input message. The following paragraphs provide design information for this CSU.

##### 4.6.24.1 CSU decode\_eom\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.24.2 CSU decode\_eom\_pdu design.

The information identified below represents the detailed design of the decode\_eom\_pdu CSU.

Syntax for invoking this function:  
output = decode\_eom\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a Recon\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Mission ID,  
Target ID,  
Mission Status Type,  
End Of Mission message string,  
Disposition Type,  
Record As Target Logical Type,  
Casualties,  
Point Number,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_eom,  
mission\_status\_enum\_str,  
disposition\_enum\_str,  
logic\_enum\_str.

**Library routines:**

sprintf,  
strcat,  
strncat.

**CSUs:**

puttogether.

**h. Logic flow.**

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

append the Mission ID title, the Mission ID string, and a carriage return to ret\_str;

append the Target ID title, the Target ID string, and a carriage return to ret\_str;

if Mission Status Type is greater than or equal to zero and it is less than the number of possible Mission Status Type.

decode Mission Status Type;

else

decode Mission Status Type with an invalid\_entry string;

invoke puttogether() to append the Mission Status title, the decoded Mission Status Type string and a carriage return to ret\_str;

append a carriage return to ret\_str;

append the "\*" \* END OF MISSION \*" message at the center of a line and two carriage returns to ret\_str;

if Disposition Type is greater than or equal to zero and it is less than the number of possible Disposition Type.

decode Disposition Type;

else

decode Disposition Type with an invalid\_entry string;

invoke puttogether() to append the Disposition title, the decoded Disposition Type string and a carriage return to ret\_str;

if Record As Target Type is greater than or equal to zero and it is less than the number of possible Logical Type.

decode Record As Target Type;

else

decode Record As Target Type with an invalid\_entry string;

invoke puttogether() to append the Record As Target title, the decoded Record As Target logical Type string and a carriage return to ret\_str;



convert Casualties Number into a string;  
invoke puttogether() to append the Casualties title, the converted Casualties Number string, and a carriage return to ret\_str;

append the Point Number title, the Point Number string, and two carriage returns to ret\_str;

append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.25 CSU decode\_status\_pdu

The decode\_status\_pdu CSU decodes the Status input message. The following paragraphs provide design information for this CSU.

##### 4.6.25.1 CSU decode\_status\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.25.2 CSU decode\_status\_pdu design.

The information identified below represents the detailed design of the decode\_status\_pdu CSU.

Syntax for invoking this function:  
output = decode\_status\_pdu( pdu);

- a. Input/output data elements.

##### INPUTS:

" pdu" is a Status\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

"i, j" are loop counters.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Fuel Status,  
Failed Equipments Type,  
Hell Fires,  
Stingers,  
Rockets,  
Rounds,  
Request type,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_status,  
fail\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

None.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

convert Fuel into a string with unit "LBS" and a carriage return to ret\_str;

repeat for three failed equipments

if Failed Equipment Type is greater than or equal to zero and is less than the number of possible Failed Equipment.

decode Failed Equipment Type;

else

decode Failed Equipment with an invalid\_entry string;

append the Failed Equipment title, the decoded Failed Equipment Type string, and a carriage return to ret\_str;

convert Hell Fires into a string;

append the Hell Fires title, the converted Hell Fires string, and a carriage return to ret\_str;

convert Stingers into a string;

append the Stingers title, the converted Stingers string, and a carriage return to ret\_str;

convert Rockets into a string;

append the Rockets title, the converted Rockets string, and a carriage return to ret\_str;

convert Rounds into a string;

append the Rounds title, the converted Rounds string, and a carriage return to ret\_str;

if Request Type is greater than or equal to zero and is less than the number of possible Request Type

decode Request Type;

else

decode Request Type with an invalid\_entry string;

append the Request Type title, the decoded Request Type string, and a carriage return to ret\_str;

append two carriage returns to ret\_str( two blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.26 CSU decode\_request\_pdu

The decode\_request\_pdu CSU decodes the Request input message. The following paragraphs provide design information for this CSU.

##### 4.6.26.1 CSU decode\_request\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.26.2 CSU decode\_request\_pdu design.

The information identified below represents the detailed design of the decode\_request\_pdu CSU.

Syntax for invoking this function:

output = decode\_request\_pdu( pdu);

- a. Input/output data elements.

###### INPUTS:

"pdu" is a Request\_type pointer containing the input message to be decoded.

###### OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

- b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Report Type,

Recon Type,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_request,

report\_enum\_str.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str(a blank line);

if Report Type is greater than or equal to zero and is less than the  
number of possible Report Type  
decode Report Type;

else

decode Report Type with an invalid\_entry string;

invoke puttogether to append the Report Type title, the decoded Report  
Type string, and a carriage return to ret\_str;

if Report type is equal to Reconnaissance

if Recon Type is greater than or equal to zero and is less  
than the number of possible Recon Type

decode Recon Type;

else

decode Recon Type with an

```
        invalid_entry string;  
        invoke puttogether to append the Recon    Type title, the  
        decoded Recon Type string,    and a carriage return to ret_str;  
    else  
        append a carriage to ret_str;  
  
    append nine carriage returns to ret_str(nine blank  
    lines);  
    append the first 32 Free Text characters and a carriage to ret_str;  
    append the next 32 Free Text characters) and a carriage to ret_str;  
    return ret_str to caller.
```

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.27 CSU decode\_nbc1\_pdu

The decode\_nbc1\_pdu CSU decodes the Nuclear, Biological, Chemical of Type one input message. The following paragraphs provide design information for this CSU.

##### 4.6.27.1 CSU decode\_nbc1\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.27.2 CSU decode\_nbc1\_pdu design.

The information identified below represents the detailed design of the decode\_nbc1\_pdu CSU.

Syntax for invoking this function:

output = decode\_nbc1\_pdu( pdu);

- a. Input/output data elements.

##### INPUTS:

" pdu" is a NBC\_type pointer containing the input message to be decoded.

**OUTPUTS:**

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Description Type,  
Burst Type,  
Delivered By,  
Cloud Height Unit,  
Cloud Height,  
Cloud Description,  
Flash Bang Time,  
Start Time,  
Stop Time,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types -

character pointer:

decode\_nbc1,  
nbc\_desc\_enum\_str,  
burst\_enum\_str,  
delivery\_enum\_str,  
cloud\_hgt\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

decode\_time,  
puttogether.

h. Logic flow.

set ret\_str to null;

append a carriage return to ret\_str;

append the ">>>> NUCLEAR BLAST <<<<" message string at the center of a line and a carriage return to ret\_str;

if Description is greater than or equal to zero and is less than the number of possible Description Type.

decode Description Type into a string;

else

decode Description Type with an invalid\_entry string;

invoke puttogether() to append the Description Type title, the decoded Description Type string and a carriage return to ret\_str;

if Burst Type is greater than or equal to zero and is less than the number of possible Burst Type.

decode Burst Type into a string;

else

decode Burst Type with an invalid\_entry string;

invoke puttogether() to append the Burst Type title, the decoded Burst Type string and a carriage return to ret\_str;

if Delivered By is greater than or equal to zero and is less than the number of possible Delivered By Type.

decode Delivered By Type into a string;

else

decode Delivered By Type with an invalid\_entry string;

invoke puttogether() to append the Delivered By Type title, the decoded Delivered By Type string and a carriage return to ret\_str;

if Cloud Height Unit Type is greater than or equal to zero and is less than the number of possible Cloud Height Unit Type.

decode Cloud Height Unit Type into a string;

else

decode Cloud Height Unit Type with an invalid\_entry string;

invoke puttogether() to append the Cloud Height Unit Type title, the decoded Cloud Height Unit Type string and a carriage return to ret\_str;

convert the Cloud Height into a string;

append the Cloud Height title, the converted Cloud Height string, and a carriage return to ret\_str;



append the Cloud Description title, the Cloud Description string, and a carriage return to ret\_str;

convert the Flash Bang Time into a string with unit "SECONDS";  
append the Flash Bang Time title, the converted Flash Bang Time string, and a carriage return to ret\_str;

invoke decode\_time to decode the Start Time;  
append the Start Time title, the decoded Start Time, and a carriage return to ret\_str;

invoke decode\_time to decode the Stop Time;  
append the Stop Time title, the decoded Stop Time, and a carriage return to ret\_str;

append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.28 CSU decode\_nbc4\_pdu

The decode\_nbc4\_pdu CSU decodes the Nuclear, Biological, Chemical of Type four input message. The following paragraphs provide design information for this CSU.

##### 4.6.28.1 CSU decode\_nbc4\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.28.2 CSU decode\_nbc4\_pdu design.

The information identified below represents the detailed design of the decode\_nbc4\_pdu CSU.

Syntax for invoking this function:  
output = decode\_nbc4\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a NBC\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Description Type,  
Burst Type,  
Delivered By,  
Cloud Height Unit,  
Cloud Height,  
Cloud Description,  
Dose Rate,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_nbc4,  
nbc\_desc\_enum\_str,  
burst\_enum\_str,

delivery\_enum\_str,  
cloud\_hgt\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

set ret\_str to a null string;

append a carriage return to ret\_str;

append the "\* CHEMICAL OR BIOLOGICAL \*" message string at the  
center of a line and two carriage returns to ret\_str;

if Description is greater than or equal to zero and is less than the  
number of possible Description Type.

decode Description Type into a string;

else

decode Description Type with an invalid\_entry string;

invoke puttogether() to append the Description Type title, the decoded  
Description Type string and a carriage return to ret\_str;

if Burst Type is greater than or equal to zero and is less than the  
number of possible Burst Type.

decode Burst Type into a string;

else

decode Burst Type with an invalid\_entry string;

invoke puttogether() to append the Burst Type title, the decoded Burst  
Type string and a carriage return to ret\_str;

if Delivered By is greater than or equal to zero and is less than the  
number of possible Delivered By Type.

decode Delivered By Type into a string;

else

decode Delivered By Type with an invalid\_entry string;

invoke puttogether() to append the Delivered By Type title, the  
decoded Delivered By Type string and a carriage return to ret\_str;

if Cloud Height Unit Type is greater than or equal to zero and is less  
than the number of possible Cloud Height Unit Type.

decode Cloud Height Unit Type into a string;

else

decode Cloud Height Unit Type with an invalid\_entry string;

invoke puttogether() to append the Cloud Height Unit Type title, the decoded Cloud Height Unit Type string and a carriage return to ret\_str;

convert the Cloud Height into a string;  
append the Cloud Height title, the converted Cloud Height string, and a carriage return to ret\_str;

append the Cloud Description title, the Cloud Description string, and a carriage return to ret\_str;

convert the Dose Rate into a string;  
append the Dose Rate title, the converted Dose Rate string, and a carriage return to ret\_str;

append two carriage returns to ret\_str(two blank lines);  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None

#### 4.6.29 CSU decode\_nbc5\_pdu

The decode\_nbc5\_pdu CSU decodes the Nuclear, Biological, Chemical of Type five input message. The following paragraphs provide design information for this CSU.

##### 4.6.29.1 CSU decode\_nbc5\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.29.2 CSU decode\_nbc5\_pdu design.

The information identified below represents the detailed design of the decode\_nbc5\_pdu CSU.

Syntax for invoking this function:  
output = decode\_nbc5\_pdu( pdu);

a. Input/output data elements.

INPUTS:

"pdu" is a NBC\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements. None.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append five carriage returns to ret\_str.

append the following fields to ret\_str in order:

message string;

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_nbc5.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

None.

h. Logic flow.

place a null to ret\_str;

append five carriage returns to ret\_str(five blank lines);

append the "NBC NEGATIVE" message string at the center of a line  
and a carriage return to ret\_str;

append six carriage returns to ret\_str(six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None

#### 4.6.30 CSU decode\_miji\_pdu

The decode\_miji\_pdu CSU decodes the Meaconing, Intrusion, Jamming Interface input message. The following paragraphs provide design information for this CSU.

##### 4.6.30.1 CSU decode\_miji\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.30.2 CSU decode\_miji\_pdu design.

The information identified below represents the detailed design of the decode\_miji\_pdu CSU.

Syntax for invoking this function:  
output = decode\_miji\_pdu( pdu);

- a. Input/output data elements.

###### INPUTS:

"pdu" is a MIJI\_type pointer containing the input message to be decoded.

###### OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

- b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

"i, j" are integer loop counters.  
"temp1" is a 64 bit float storage.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Affected Radio Frequency,  
Programmed Frequency1,  
Start Time,  
Stop Time,  
Description,  
Percent Lost,  
Type,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types -  
character pointer:

decode\_miji,  
description\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat,  
for-loop.

CSUs:

freq\_hertz,  
decode\_time,  
puttogether.

h. Logic flow.

place a null to ret\_str;

append a carriage return to ret\_str(a blank line);

invoke freq\_hertz to decode the Affected Radio Frequency to its  
appropriate unit "kilo, Mega, or Giga" Hertz.

append the Affected Radio Frequency title, the decoded Affected Radion Frequency, and a carriage return to ret\_str;

repeat four times

    invoke freq\_hertz to decode the           Programmed Frequency  
    to its appriorate unit           "kilo, Mega, or Giga" Hertz.

    append the Programmed Frequency title,    the    decoded  
    Programmed Frequency, and    a carriage return to ret\_str;

invoke decode\_time to decode the Start Time;

append the Start Time title, the decode Start Time, and a carriage  
return to ret\_str;

invoke decode\_time to decode the Stop Time;

append the Stop Time title, the decode Stop Time, and a carriage return  
to ret\_str;

if Description is greater than or equal to zero and is less than the  
number of possible Description Type.

    decode Description Type into a string;

else

    decode Description Type with an invalid\_entry string;

append the Description title, the decoded Description Type, and a  
carriage return to ret\_str;

convert Percent Lost into a string with unit"PERCENT";

append the Percent Lost title, the converted Percent Lost string, and a  
carriage return to ret\_str;

if Type is greater than or equal to zero and is less than the number of  
possible Type.

    decode Type into a string;

else

    decode Type with an invalid\_entry string;

append the Type title, the decoded Type, and two carriage returns to  
ret\_str;

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i.     Data structures. None

j.     Local data files or database. None

k.     Limitations. None



#### 4.6.31 CSU decode\_dnav\_pdu

The decode\_dnav\_pdu CSU decodes the Downed Air Vehicle Report input message. The following paragraphs provide design information for this CSU.

##### 4.6.31.1 CSU decode\_dnav\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.31.2 CSU decode\_dnav\_pdu design.

The information identified below represents the detailed design of the decode\_dnav\_pdu CSU.

Syntax for invoking this function:

```
output = decode_dnav_pdu( pdu);
```

a. Input/output data elements.

INPUTS:

" pdu" is a DNAV\_type pointer containing the input message to be decoded.

OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. None

d. Algorithms.

store a null in the first byte of ret\_str.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Aircraft Type,

Aircraft Status,

Pilot Status,  
Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

The following provide the string values of "Enumerated" types - character pointer:

decode\_dnav,  
aircraft\_enum\_str,  
aircraft\_status\_enum\_str,  
pilot\_status\_enum\_str.

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

puttogether.

h. Logic flow.

initialize ret\_str to a null string;

append a carriage return to ret\_str(a blank line);

if Aircraft Type is greater than or equal to zero and it is less than the number of possible Aircraft Type.

decode Aircraft Type;

else

decode Aircraft Type with an invalid\_entry string;

invoke puttogether() to append the Aircraft Type title, the decoded Aircraft Type string and a carriage return to ret\_str;

if Aircraft Status is greater than or equal to zero and is less than the number of possible Aircraft Status.

decode Aircraft Status;

else

decode Aircraft Status with an invalid\_entry string;

invoke puttogether() to append the Aircraft Status title, the decoded Aircraft Status string and a carriage return to ret\_str;

if Pilot Status is greater than or equal to zero and is less than the number of possible Pilot Status.

decode Pilot Status;

else  
    decode Pilot Status with an invalid\_entry string;  
    invoke puttogether() to append the Pilot Status title, the decoded Pilot  
    Status string and a carriage return to ret\_str;

append eight carriage returns to ret\_str(eight blank lines);  
append the first 32 Free Text characters and a carriage to ret\_str;  
append the next 32 Free Text characters) and a carriage to ret\_str;  
return ret\_str to caller.

- i.     Data structures. None
- j.     Local data files or database. None
- k.     Limitations. None

#### 4.6.32 CSU decode\_movcmd\_pdu

The decode\_movcmd\_pdu CSU decodes the Move Command input message.  
The following paragraphs provide design information for this CSU.

##### 4.6.32.1 CSU decode\_movcmd\_pdu requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.32.2 CSU decode\_movcmd\_pdu design.

The information identified below represents the detailed design of the decode\_movcmd\_pdu CSU.

Syntax for invoking this function:

output = decode\_movcmd\_pdu( input message pointer of Move\_type);

- a.     Input/output data elements.

##### INPUTS:

"pdu" is a Move\_type pointer containing the input message to be decoded.

##### OUTPUTS:

"ret\_str" is a character buffer of size 18x36 containing the decoded message string to be returned.

b. Local data elements.

"temp\_str" is a static unsigned char buffer of size 36 that contains the temporary decoded message.

"temp" is a temporary storage that contains the integer value of a message field.

c. Interrupts and signals. Noned. Algorithms.

initialize ret\_str to null.

append a carriage return to ret\_str.

decode and append the following fields to ret\_str in order:

Target ID,

Task,

Who,

When,

Where,

Free Text Annotation String.

\* append a carriage return to ret\_str at the end of each field.

e. Error handling. Nonef. Data conversion. Noneg. Use of other elements.

The following provide the string values of "Enumerated" types -

character pointer:

decode\_movcmd,

task\_enum\_str,

who\_enum\_str,

when\_enum\_str.

Library routines:

sprintf,

strcat,

strncat.

CSUs:

puttogether.

h. Logic flow.

initialize ret\_str to a null string;

append a carriage return to ret\_str;

append the Targe ID title, the Targe ID string, and a carriage return to ret\_str;

if Task is greater than or equal to zero and it is less than the number of possible Task.

decode Task;

else

decode Task with an invalid\_entry string;

invoke puttogether() to append the Task title, the decoded Task string and a carriage return to ret\_str;

if Who is greater than or equal to zero and is less than the number of possible Who list.

decode Who;

else

decode Who with an invalid\_entry string;

invoke puttogether() to append the Who title, the decoded Who string and a carriage return to ret\_str;

if When is greater than or equal to zero and is less than the number of possible When list.

decode When;

else

decode When with an invalid\_entry string;

invoke puttogether() to append the When title, the decoded When string and a carriage return to ret\_str;

append the Where title, the Where string, and a carriage return to ret\_str;

append six carriage returns to ret\_str(six blank lines);

append the first 32 Free Text characters and a carriage to ret\_str;

append the next 32 Free Text characters) and a carriage to ret\_str;

return ret\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.33 CSU puttogether

The puttogether CSU appends message title and decoded message into a temporary character string buffer. The following paragraphs provide design information for this CSU.

**4.6.33.1 CSU puttogether requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.6.33.2 CSU puttogether design.**

The information identified below represents the detailed design of the puttogether CSU.

Syntax for invoking this function:

puttogether(decoded message, return buffer, message header);

a. Input/output data elements.

INPUTS:

"temp\_str" is an unsigned character pointer containing the decoded message.

"msg" is an unsigned character pointer containing the message title.

OUTPUTS:

"ret\_str" is an unsigned character pointer containing the return buffer.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

data elements:

None

Library routines:

strcat.

CSUs:

None

- h. Logic flow.  
append message title, decoded message, and a carriage return to ret\_str;  
return ret\_str to caller.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.34 CSU decode\_time

The decode\_time CSU decodes DTG time. The following paragraphs provide design information for this CSU.

##### 4.6.34.1 CSU decode\_time requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.34.2 CSU decode\_time design.

The information identified below represents the detailed design of the decode\_time CSU.

Syntax for invoking this function:

decode\_time(return buffer, DTG time);

- a. Input/output data elements.

##### INPUTS:

"dtg" is a DTG\_type pointer containing the input DTG time to be decoded.

##### OUTPUTS:

"temp\_str" is an unsigned character buffer of containing the decoded time string to be returned.

- b. Local data elements.

"month" is an integer storage.

"temp\_str2" is a static unsigned character buffer of size 36 containing temporary Month string.

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

None

Library routines:

sprintf,  
strcat,  
strncat.

CSUs:

None.

h. Logic flow.

initialize temp\_str to a null string and temp\_str2 to initialize the string buffer;

append Day and a space to temp\_str;

append Hour, Minute, and a space to temp\_str;

decode Month into its appropriate interpretation "JAN, FEB, MAR, ... DEC".

append the decoded Month string and a space to temp\_str;

append the last two characters of Year string to temp\_str;

return temp\_str to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

#### 4.6.35 CSU freq\_hertz

The freq\_hertz CSU decodes the the input frequency into its proper unit. The following paragraphs provide design information for this CSU.



**4.6.35.1 CSU freq\_hertz requirements.**

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.6.35.2 CSU freq\_hertz design.**

The information identified below represents the detailed design of the freq\_hertz CSU.

Syntax for invoking this function:

freq\_hertz(input frequency, return buffer);

a. Input/output data elements.

INPUTS:

"temp1" is a 64 bit float integer containing the input frequency to be decoded.

OUTPUTS:

"temp\_str" is a character buffer of size 36 containing the decoded frequency string to be returned.

b. Local data elements.

"temp, temp2" are temporary 64 bit float storage;

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

None.

Library routines:

if - else,

sp. atf.

CSUs:

None.

- h. Logic flow.  
set temp to zero;  
if input frequency >= a kilo and is < a million, convert it to kiloHertz;  
else if input frequency >= a million and is < a billion, convert it to MegaHertz;  
else if input frequency >= a billion, convert it to GigaHertz;  
else append the Hertz unit to input frequency;  
place the decoded input frequency to temp\_str;  
return temp\_str to caller.
- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.36 CSU sprintf\_UTM

The sprintf\_UTM CSU decodes the input DTG time. The following paragraphs provide design information for this CSU.

##### 4.6.36.1 CSU sprintf\_UTM requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

##### 4.6.36.2 CSU sprintf\_UTM design.

The information identified below represents the detailed design of the sprintf\_UTM CSU.

Syntax for invoking this function:

output = sprintf\_UTM(address of the input location);

- a. Input/output data elements.

##### INPUTS:

"utm" is a UTM\_type pointer containing the input location to be decoded.

##### OUTPUTS:

"ret\_str" is a static character buffer of size 80 containing the decoded UTM string to be returned.

- b. Local data elements. None
- c. Interrupts and signals. None
- d. Algorithms. None
- e. Error handling. None
- f. Data conversion. None

g. Use of other elements.

Data elements:

None.

Library routines:

strncpy.

CSUs:

None.

h. Logic flow.

place the two Sector Letters and a space to ret\_str;

append the first four Easting characters and a space to ret\_str;

append the first four Northing characters and a null character to ret\_str;

return ret\_str to caller.

- i. Data structures. None
- j. Local data files or database. None
- k. Limitations. None

#### 4.6.37 CSU sprintf\_speed

The sprintf\_speed CSU decodes the input Speed into its string representation. The following paragraphs provide design information for this CSU.

##### 4.6.37.1 CSU sprintf\_speed requirements.

This CSU satisfies internally derived contractor requirements to detail an implementation of a Digital Message Communications System compliant with customer directives to provide a man-machine-interface (MMI) and system behavior which emulate the digital message communications subfunction of a helicopter mission equipment package.

**4.6.37.2 CSU sprintf\_speed design.**

The information identified below represents the detailed design of the sprintf\_speed CSU.

Syntax for invoking this function:

output = sprintf\_speed(address of the input speed);

a. Input/output data elements.

INPUTS:

"speed" is a Speed\_type pointer containing the input Speed to be decoded.

OUTPUTS:

"speed\_buffer" is a static character buffer of size 20 containing the decoded Speed string to be returned.

b. Local data elements. None

c. Interrupts and signals. None

d. Algorithms. None

e. Error handling. None

f. Data conversion. None

g. Use of other elements.

Data elements:

None.

Library routines:

if - else

sprintf.

CSUs:

None.

h. Logic flow.

decode the unit of Speed type;

append Speed value, a space, and the decoded Speed unit string to speed\_buffer;

return speed\_buffer to caller.

i. Data structures. None

j. Local data files or database. None

k. Limitations. None

## 5. DMCC Global Data Elements

In writing C programs, great use is made of 'header' files. These are include files that contain the information necessary for multiple modules to treat data in the same way. For this CSCI, the header files contain structure definitions that allow different functions to fill selected data entities within the structures, modify or use all or parts of those structures, or pass data the data structures between functions to allow for the ordered manipulation of that data. Also within the header files are case sensitive statements that control how data is accessed or processed. They are delineated through the use of 'define' statements. Most of the time, these items are flags used for comparison within the functions. They can also be used as constants for array sizing. At compile time, the compiler exchanges the statement for the value that it is equated with. For example, if the header file "assoc\_PDU.h" is included into module X, then at compile time if anywhere within the source code of module X the fragment "ASSOC\_DATAGRAM" is found, the compiler will replace the fragment with the value "1" because in the header assoc\_PDU.h is found the statement '#define ASSOC\_DATAGRAM 1'. Each header file may be described in 3 parts. Part 1 of the following tables will contain the *structure* definitions, part 2 of the associated tables containing the 'define' statements, and part 3 (where applicable) will dimension variables, initialize variables, or prototype functions.

For this CSCI there are 2 uses for the header files. Those listed in section 5.1.x are for system use, while those listed in section 5.2.x are for the General User Interface (GUI) use.

### 5.1 System Header files

#### 5.1.1 assoc\_PDU.h

This header file provides the data structures and definitions needed for the decode\_pkt modules used to decode the incoming message PDU's.

5.1.1.1.1 assoc\_PDU structure assocPDU\_type

Table 4 Structure assocPDU\_Type

Item Name	Type	bytes	Description
Version, PDUKind	Integer	2	Version - bits 0:3, PDUKind bits 4:7
DataLength	Char	1	
MulticastGroup	Char	1	
Protocol	Char	1	
OriginatorSite	Integer	2	
OriginatorSimulator	Integer	2	
ResponderSite	Integer	2	
ResponderSimulator	Integer	2	
TransactionID	Integer	2	
Unused	Integer	2	

5.1.1.1.2 assoc\_PDU structure assocPDUWithLLCHeaderType

Table 5 assocPDUWithLLCHeaderType

Item Name	Type	bytes	Description
LLCSubLayerHeader	Char array	8	
AssocPDU	Structure	17	AssocPDU_type structure
ExtraPadding	Char array	32	

5.1.1.1.3 assoc\_PDU structure ActivateRequestEntry

Table 6 assocPDU structure ActivateRequestEntry

Item Name	Type	bytes	Description
AssocPDUWithLLCHeader	Structure	57	AssocPDUWithLLCHeaderType
Valid	Char	1	
MorePadding	Char array	7	

5.1.1.1.4 assoc\_PDU structure ActivateRequestBufferStructure

Table 7 assocPDU structure ActivateRequestBufferStructure

Item Name	Type	bytes	Description
ActivateRequest	Struc array	1300	array of ActivateRequestEntry types

## 5.1.1.2 assoc\_PDU.h define statements

Table 8 assoc\_PDU.h define statements

Name	Value	Description
ASSOC_DATA_LENGTH	1	offset pointer within PDU
ASSOC_MULTICAST_GROUP	2	offset pointer within PDU
ASSOC_PROTOCOL	3	offset pointer within PDU
ASSOC_VERSION_AUG_89	1	values for insertion/comparison in PDU
ASSOC_VERSION_JAN_90	2	values for insertion/comparison in PDU
ASSOC_UNKNOWN_KIND	0	values for insertion/comparison in PDU
ASSOC_DATAGRAM	1	values for insertion/comparison in PDU
ASSOC_REQUEST	2	values for insertion/comparison in PDU
ASSOC_RESPONSE	3	values for insertion/comparison in PDU
ASSOC_PADDING	4	values for insertion/comparison in PDU
ASSOC_LAST_KIND	4	values for insertion/comparison in PDU
ASSOC_MULTIPLIER	8	values for insertion/comparison in PDU
ASSOC_BROADCAST	0	values for insertion/comparison in PDU
ASSOC_SIMULATION	1	values for insertion/comparison in PDU
FORT_RUCKER	4	values for insertion/comparison in PDU
ASSOC_DIGITAL_MSG	0xD0	values for insertion/comparison in PDU
OUR_SIMULATION_NUM	96	values for insertion/comparison in PDU
ASSOC_PDUKINDMASK	0xF0	values for insertion/comparison in PDU
ASSOC_PDU_MIN_LEN	46	values for insertion/comparison in PDU
ASSOC_VER_PADKIND	0x24	values for insertion/comparison in PDU
DIS_NET	0	PDU processing codes
SIM_NET	1	PDU processing codes
NEW_ALPDU	0	PDU processing codes
CONT_ALPDU	1	PDU processing codes
END_ALPDU	2	PDU processing codes
NO_CONFIRM	0	PDU processing codes
CONFIRM	1	PDU processing codes
NO_ASSOC_LAYER	0	PDU processing codes
UDP_PORT	0x4128	Network Interface codes
HOST_NAME	adst5	Network Interface codes
SNAP_LENGTH	8	Network Interface codes
MAX_ACTIVATE_REQUESTS	20	max activate requests that can be stored
ACTIVATE_REQUEST_SHMEM_KEY	96161	
PADDING_FOR_ALPDU	32	number of bytes to complete PDU



## 5.1.2 databases.h

This header file provides the data structures needed by the database storage functions to maintain operating system data for later use.

5.1.2.1.1 databases structure IndirectFireDataBase\_typeTable 9 structure InderectFireDataBase\_type

Item Name	Type	bytes	Description
valid	Char	1	
vehicleID	Structure	6	VehicleID_type structure - 5.1.26
velocity	Structure	12	VelocityVector_type - 5.1.26
muzzle	Structure	24	WorldCoordinates_type - 5.1.26
projectileID	Structure	6	VehicleID_type structure - 5.1.26

5.1.2.1.2 databases structure DetonationDataBase\_typeTable 10 structure DetonationDataBase\_type

Item Name	Type	bytes	Description
valid	Char	1	
Location	Structure	24	WorldCoordinates_type - 5.1.26

5.1.2.1.3 databases structure VehicleAppearanceDataBase\_typeTable 11 structure VehicleAppearanceDataBase\_type

Item Name	Type	bytes	Description
valid	Char	1	
Change	Char	1	

5.1.2.1.4 databases structure StateDataBase\_type

Table 12 structure StateDataBase\_type

Item Name	Type	bytes	Description
state	structure		VehicleAppearanceVariant_type-5.1.26
DRA	Structure	40	DeadReckoning_type
valid	Char	1	new information flag
do_dr	Char	1	Dead Reckoning ON flag
time	Structure	4	TimeStamp_type
inlist	Char	1	in linked list flag
send_time	Double	8	time of rebroadcast
interval_time	Double	8	rebroadcast period
state_time	Double	8	Time of last state
X0	Double	8	Initial X position
Y0	Double	8	Initial Y position
Z0	Double	8	Initial Z position
VX0	Float	4	Initial X velocity
VY0	Float	4	Initial Y velocity
VZ0	Float	4	Initial Z velocity
AX0	Float	4	Initial X acceleration
AY0	Float	4	Initial Y acceleration
AZ0	Float	4	Initial Z acceleration
yaw0	Float	4	Initial X yaw
pitch0	Float	4	Initial Y pitch
roll0	Float	4	Initial Z roll
yaw0dot	Float	4	Initial X yaw rate
pitch0dot	Float	4	Initial Y pitch rate
roll0dot	Float	4	Initial Z roll rate
previous	Integer	4	last element in list
next	Integer	4	next element in list

5.1.2.1.5 databases structure SIMActivateRequestDataBase\_type

Table 13 structure SIMActivateRequestDataBase\_type

Item Name	Type	bytes	Description
valid	Char	1	
unit	Structure	20	OrganizationalUnit_type - 5.1.26

## 5.1.2.2 databases.h define statements

Table 14 databases.h define statements

Name	Value
MAX_DATAGRAMS	20
BUFF_SIZE	1500
PDU_NAME_NUM_OF_CHARS	40
MAX_ENTITIES	256
MAX_EVENTS	16
ERROR	-1
GE_Fulda_Gap	0
GE_Hunter_Liggett	1

## 5.1.3 declare\_netif.h

This header file declares and dimensions variables to be used by the network interface software modules. Sizes are specified by define statements in databases.h (5.1.2.2) and netif.h (5.1.24.2). Tables 5.1.3.1 and 5.1.3.2 are not applicable.

## 5.1.3.3 declare\_netif dimension statements

Table 15 declare\_netif dimension statements

Name	Type	Bytes
selection	Char	4
frame_buff[MAX_DATAGRAMS+1][BUFF_SIZE]	Char array	31500
discard_buff[BUFF_SIZE]	Char array	1500
interface_na[MAX_IFNAME_SIZE]	Char array	10
SIM_interface_name[MAX_IFNAME_SIZE]	Char array	10
DIS_interface_name[MAX_IFNAME_SIZE]	Char array	10
init_DIS_tx_netif (initialized to FALSE)	Integer	4
init_SIM_tx_netif (initialized to FALSE)	Integer	4
DIS_rx_index (initialized to 0)	Integer	4
DIS_rx_etherFd	Integer	4
DIS_rx_sockFd	Integer	4
DIS_tx_etherFd	Integer	4
DIS_tx_sockFd	Integer	4
DIS_currentPDUNumber (initialized to 0)	Integer	4
SIM_rx_index (initialized to 0)	Integer	4
SIM_rx_etherFd	Integer	4
SIM_tx_etherFd	Integer	4
SIM_currentPDUNumber (initialized to 0)	Integer	4
SIM_rx_sockFd	Integer	4
SIM_tx_sockFd	Integer	4

## 5.1.4 declare\_vars.h

This header file declares and dimensions variables to create the actual global shared memory variables. Table 5.1.4.2 is not applicable.

## 5.1.4.1 declare\_vars structures

Table 16 declar\_vars structures

Structure Name	bytes	Structure Type
dummy_DRA	40	DeadReckoning_type
DetDB [MAX_ENTITIES]		DetonationDataBase_type
StateDB [MAX_ENTITIES]		StateDataBase_type
IndFireDB [MAX_ENTITIES][MAX_EVENTS]		IndirectFireDataBase_type
VehAppDB [MAX_ENTITIES]		VehicleAppearanceDataBase_type
SActReqDB [MAS_ENTITIES]		SIMActivateRequestDataBase_type
dummy_TimeStamp		TimeStamp_type
Startup_Time		timeval
Time_Zone		timezone
Startup_TimeStamp		TimeStamp_type

## 5.1.4.3 declare\_vars dimension statements

Table 17 declar\_vars dimension statements

Name	Type	Bytes
My_site	Integer	2
DIS_netif_pid	Integer	4
DIS_sigrec (initialized to 0)	Integer	4
SIM_sigxmt (initialized to 0)	Integer	4
SIM_next_frame (initialized to 0)	Integer	4
SIM_send_frame (initialized to 0)	Integer	4
SIM_rm_left	Integer	4
DIS_remainder	Integer	4
DIS_consumed (initialized to 0)	Integer	2
SIM_out_ALPDU_kind	Char	1
DIS_PDU_in	Char pointer	4
SIM_PDU_out	Char pointer	4
DIS_rxbuf [BUFF_SIZE]	Char array	1500
SIM_txbuf [MAX_DATAGRAMS][BUFF_SIZE]	Char array	30000
SIM_netif_pid	Integer	4
SIM_sigrec (initialized to 0)	Integer	4
DIS_sigxmt (initialized to 0)	Integer	4
DIS_next_frame (initialized to 0)	Integer	4
DIS_send_frame (initialized to 0)	Integer	4
DIS_rm_left	Integer	2
SIM_remainder	Integer	4
SIM_consumed (initialized to 0)	Integer	2

DIS_out_ALPDU_kind	Char	1
SIM_PDU_in	Char pointer	4
DIS_PDU_out	Char pointer	4
SIM_rxbuf [BUFF_SIZE]	Char	1500
DIS_remainder	Integer	4
DIS_consumed (initialized to 0)	Integer	2
SIM_out_ALPDU_kind	Char	1
DIS_PDU_in	Char pointer	4
SIM_PDU_out	Char pointer	4
SIM_rxbuf [BUFF_SIZE]	Char array	1500
DIS_txbuf [MAX_DATAGRAMS][BUFF_SIZE]	Char array	30000
DIS_Assoc_Layer ( initialized to TRUE)	Char	1
SIM_Assoc_Layer ( initialized to TRUE)	Char	1
pass_all ( initialized to TRUE)	Char	1
Xlat_DMC ( initialized to TRUE)	Char	1
Exercise_ID	Char	1
Protocol_ID	Char	1
Battle_Scheme	Char	1
logger_datafile	File pointer	4
send_infile	Char array	80
ASSOC_hdr_leng	Const Char array	5
SIMPDU_name [NUMBER_SIM_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	800
DISPDU_name [NUMBER_DIS_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	760
DMCPDU_name [NUMBER_DMC_PDUS][PDU_NAME_NUM_OF_CHARS]	Const String array	600
SIMPDU_max_delta [NUMBER_SIM_PDUS]	Integer	40
SIMPDU_base_leng [NUMBER_SIM_PDUS]	Integer	40
DISPDU_max_delta [NUMBER_DIS_PDUS]	Integer	38
DISPDU_base_leng [NUMBER_DIS_PDUS]	Integer	38
DMCPDU_base_leng [NUMBER_DMC_PDUS]	Integer	30
SIMPDU_xlates_to [NUMBER_SIM_PDUS]	Integer	40
DISPDU_xlates_to [NUMBER_DIS_PDUS]	Integer	38
PDU_error [NUMBER_PDU_ERRORS][PDU_NAME_NUM_OF_CHARS]	Const String array	1000
terrain_database	Integer	4
ft_per_asec	Double	8
deg_to_rad	Double	8
meters_to_feet	Double	8
degrees_to_asec	Double	8
feet_to_meters	Double	8
pnumber	Integer	4
GE_DB_origin_lat	Double	8
GE_DB_origin_lon	Double	8
GE_DB_SW_corner_lat	Double	8

GE_DB_SW_corner_lon	Double	8
GE_DB_NE_corner_lat	Double	8
GE_DB_NE_corner_lon	Double	8
lat_ref	Double	8
lon_ref	Double	8
sw_latitude_asec	Double	8
sw_longitude_asec	Double	8
coslat	Double	8
xoffset	Double	8
yoffset	Double	8
Fulda_DB_origin_lat	Double	8
Fulda_DB_origin_lon	Double	8
Fulda_DB_SW_corner_lat	Double	8
Fulda_DB_SW_corner_lon	Double	8
Fulda_DB_NE_corner_lat	Double	8
Fulda_DB_NE_corner_lon	Double	8
lat_ref_fulda	Double	8
lon_ref_fulda	Double	8
sw_latitude_asec_fulda	Double	8
sw_longitude_asec_fulda	Double	8
coslat_fulda	Double	8
xoffset_fulda	Double	8
yoffset_fulda	Double	8
FGECCX	Double array	80
FGECCY	Double array	80
FGECCZ	Double array	80
fgegcX	Double	8
fgegcY	Double	8
fgegcZ	Double	8
fgegcXO	Double	8
fgegcYO	Double	8
fgegcZO	Double	8
FGCCGX	Double array	80
FGCCGY	Double array	80
FGCCGZ	Double array	80
fgcgeX	Double	8
fgcgeY	Double	8
fgcgeZ	Double	8
fgcgeXO	Double	8
fgcgeYO	Double	8
fgcgeZO	Double	8
HGECCX	Double array	80
HGECCY	Double array	80
HGECCZ	Double array	80
hgegcX	Double	8
hgegcY	Double	8
hgegcZ	Double	8
hgegcXO	Double	8
hgegcYO	Double	8

hgegcZO	Double	8
HGCGEX	Double array	80
HGCGEY	Double array	80
HGCGEZ	Double array	80
hgcgeX	Double	8
hgcgeY	Double	8
hgcgeZ	Double	8
hgcgeXO	Double	8
hgcgeYO	Double	8
hgcgeZO	Double	8
HUTMGCX	Double array	80
HUTMGCY	Double array	80
HUTMGCZ	Double array	80
hutmgcX	Double	8
hutmgcY	Double	8
hutmgcZ	Double	8
hutmgcXO	Double	8
hutmgcYO	Double	8
hutmgcZO	Double	8
HGCUTMX	Double array	80
HGCUTMY	Double array	80
HGCUTMZ	Double array	80
hgcutmX	Double	8
hgcutmY	Double	8
hgcutmZ	Double	8
hgcutmXO	Double	8
hgcutmYO	Double	8
hgcutmZO	Double	8
HUTM84GCX	Double array	80
HUTM84GCY	Double array	80
HUTM84GCZ	Double array	80
hutm84gcX	Double	8
hutm84gcY	Double	8
hutm84gcZ	Double	8
hutm84gcXO	Double	8
hutm84gcYO	Double	8
hutm84gcZO	Double	8
HGCUTM84X	Double array	80
HGCUTM84Y	Double array	80
HGCUTM84Z	Double array	80
hgcutm84X	Double	8
hgcutm84Y	Double	8
hgcutm84Z	Double	8
hgcutm84XO	Double	8
hgcutm84YO	Double	8
hgcutm84ZO	Double	8
angular_tolerance	Float	4
linear_tolerance	Float	4
minimum_timer_value	Float	4



cease_time	Float	4
rebroadcast_time	Float	4
head	Integer	4
current	Integer	4
last	Integer	4
ID_index	Integer	4
history_index	Integer	4
shmaddr	Char pointer	4
shmemptr	shmen_structure pointer	4
shmemid	Integer	4
childpid	Integer	4
extension	Integer	4
out_fp	File descriptor pointer	4
start_time	Double	8
end_time	Double	8
old_time1	Double	8
old_time2	Double	8
old_time3	Double	8
old_time4	Double	8
Netif [MAX_IFNAME_SIZE]	Char array	10
s1	Char array	100
s2	Char array	100

### 5.1.5 global\_vars.h

This header file provides the data structures and definitions needed for each module that makes use of global storage. This header includes the following header files to maintain commonality among the various functions; DMC\_PDU.h, decode\_defines.h, decode\_strs.h, decode\_list.h, and decode\_fprotos.h. These header files, included in this order, dimension and define the globally used structures and data entities.

### 5.1.6 decode\_defines.h

This header file defines the sizes of the data structures used by the Digital Message Communications Console.

#### 5.1.6.1.1 decode\_defines structure Non\_specific

Table 18 decode\_defines structure Non\_specific

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type

## 5.1.6.1.2 decode\_defines define statements

Table 19 decode\_defines define statements

Name	Value	Description
FREE_TEXT_WIDTH	26	width of Free Text line in Free Text message
NUM_HEADER_LINES	8	
NUM_LINES	15	
MAX_DMC_LINES	--	NUM_HEADER_LINES + NUM_LINES
STRING_SIZE	36	
NUM_DMC_LOGIC_TYPES	2	logic types
NUM_DMC_SPEEDS	2	speed types
NUM_DMC_TERM_TYPES	5	terminal types
NUM_DMC_PERSON_TYPES	10	person types
NUM_DMC_PRIORITY_TYPES	3	priority types
NUM_DMC_ACTIVITY_TYPES	11	activity types
NUM_DMC_TARGET_TYPES	10	target types
NUM_DMC_DIRECTION_TYPES	10	direction types
NUM_DMC_OBS_INTENTIONS	7	observer intention types
NUM_DMC_CLASSIF_TYPES	7	classification types
NUM_DMC_OBSTACLE_TYPES	10	obstacle types
NUM_DMC_BRIDGE_TYPES	11	bridge types
NUM_DMC_BRIDGE_DAMAGE_TYPES	6	bridge damage types
NUM_DMC_BRIDGE_MATERIAL_TYPES	6	bridge construction material types
NUM_DMC_ACTIVITY_LIKELY_TYPES	5	activity likely probability types
NUM_DMC_MISSION_STATUS_TYPES	5	mission status types
NUM_DMC_MISSION_TYPES	5	mission types
NUM_DMC_SHELL_TYPES	5	shell types
NUM_DMC_CONTROL_TYPES	4	control types
NUM_DMC_FUZE_TYPES	4	fuse types
NUM_DMC_TRAJECTORY_TYPES	4	trajectory types
NUM_DMC_DISPOSITION_TYPES	3	disposition types
NUM_DMC_CASUALTY_TYPES	2	casualty types
NUM_DMC_COVERAGE_TYPES	6	percent coverage types
NUM_DMC_TASK_TYPES	11	task types
NUM_DMC_WHO_TYPES	6	
NUM_DMC_WHEN_TYPES	5	
NUM_DMC_FAIL_TYPES	10	equipment failure types
NUM_DMC_REQ_TYPES	2	request types
NUM_DMC_REPORT_TYPES	7	report types
NUM_DMC_RECON_TYPES	7	recon types
NUM_DMC_NBC_DESC_TYPES	10	NBC message enumerations
NUM_DMC_BURST_TYPES	6	burst types
NUM_DMC_DELIVERY_TYPES	7	delivery types
NUM_DMC_CLOUD_HGT_TYPES	4	cloud height unit types

NUM_DMC_DESC_TYPES	8	MIJI descripton types
NUM_DMC_TYPE_TYPES	6	
NUM_DMC_VISIBILITY_TYPES	8	visibility types
NUM_DMC_RESTRICT_TYPES	10	restriction types
NUM_DMC_CLOUD_TYPES	5	cloud cover types
NUM_DMC_TURBINT_TYPES	6	turbulence intensity types
NUM_DMC_TURBFREQ_TYPES	5	turbulence frequency types
NUM_DMC_ICING_INTEN_TYPES	6	icing intensity types
NUM_DMC_ICING_TYPES	4	icing types
NUM_DMC_AIRCRAFT_TYPES	13	aircraft types
NUM_DMC_AIRCRAFT_STATUS_TYPES	5	aircraft status types
NUM_DMC_PILOT_STATUS_TYPES	6	pilot status types
MSG_LINE0	0	
MSG_LINE1	1	
MSG_LINE2	2	
MSG_LINE3	3	
MSG_LINE4	4	
MSG_LINE5	5	
MSG_LINE6	6	
MSG_LINE7	7	
MSG_LINE8	8	
MSG_LINE9	9	
MSG_LINE10	10	
MSG_LINE11	11	
MSG_LINE12	12	
MSG_LINE13	13	
MSG_LINE14	14	
CHAR_STR	4	
CHAR_STR1	8	
CHAR_STR2	16	
CHAR_PRINT	32	
FREE_TEXT_PER_LINE	26	

### 5.1.7 decode\_fprotos.h

In C, function prototyping (the defining the input arguments and resultant output from the function) is required. The compiler will check the prototyping of a function against the actual arguments used within the source code of the calling function and flag as a compiler error any incongruities. This header file prototypes all of the functions used in the decode portion of the DMC.

### 5.1.8 decode\_list.h

This header file provides the message strings that are needed by the decode modules. All of these arrays are filled within this header file to be indexed by the decode modules with pointers previously defined. Tables 5.1.8.1 and 5.1.8.2 are not defined for this header file.

## 5.1.8.3 decode\_list dimension statements

Table 20 decode\_list dimension statements

Name	Type	Bytes
decode_header_label	Const String array	288
decode_ack_label	Const String array	540
decode_freetext_value	Char array	540
decode_freetext_label	Const String array	540
decode_spot_label	Const String array	540
decode_gndroute	Const String array	540
decode_airroute	Const String array	540
decode_bridge	Const String array	540
decode_lzpz	Const String array	540
decode_bpop	Const String array	540
decode_crossing	Const String array	540
decode_repeat	Const String array	540
decode_cancel	Const String array	540
decode_check	Const String array	540
decode_cno	Const String array	540
decode_shift	Const String array	540
decode_nwmsn	Const String array	540
decode_mto	Const String array	540
decode_shot	Const String array	540
decode_splash	Const String array	540
decode_eom	Const String array	540
decode_request	Const String array	540
decode_movcmd	Const String array	540
decode_status	Const String array	540
decode_nbc1	Const String array	540
decode_nbc4	Const String array	540
decode_nbc5	Const String array	540
decode_miji	Const String array	540
decode_pirep	Const String array	540
decode_dnav	Const String array	540
decode_bda	Const String array	540
decode_month	Const String array	540
decode_subtype	Const String array	540
decode_recon_variation	Const String array	540
decode_artillery_variation	Const String array	540
decode_nbc_variation	Const String array	540

## 5.1.9 decode\_list\_proto.h

This header file prototypes the string arrays defined in `decode_list.h`, section 5.1.8, for use by the decode modules.

## 5.1.10 decode\_strs.h

This header file provides the string values for "Enumerated" character types through the use of pointers, based on sizing provided by define statements in `decode_defines.h`, section 5.1.6. As these are strings, they are variable in length. The table will indicate the number of elements in each array. Tables 5.1.10.1 and 5.1.10.2 are not applicable.

## 5.1.10.3 decode\_strs dimension statements

Table 21 decode\_strs dimension statements

Name	Type	Number
message_enum_str	Enumerated String	15
invalid_entry	Enumerated String	1
logic_enum_str	Enumerated String	2
speed_enum_str	Enumerated String	2
term_enum_str	Enumerated String	5
person_enum_str	Enumerated String	10
priority_enum_str	Enumerated String	3
target_enum_str	Enumerated String	10
activity_enum_str	Enumerated String	11
direction_enum_str	Enumerated String	10
obs_int_enum_str	Enumerated String	7
classif_enum_str	Enumerated String	7
obstacle_enum_str	Enumerated String	10
bridge_enum_str	Enumerated String	11
bridge_damage_enum_str	Enumerated String	6
bridge_material_enum_str	Enumerated String	6
activity_likely_enum_str	Enumerated String	5
mission_status_enum_str	Enumerated String	5
mission_enum_str	Enumerated String	5
shell_enum_str	Enumerated String	7
control_enum_str	Enumerated String	4
fuze_enum_str	Enumerated String	4
trajectory_enum_str	Enumerated String	4
disposition_enum_str	Enumerated String	3
casualty_enum_str	Enumerated String	2
percent_cover_enum_str	Enumerated String	6
task_enum_str	Enumerated String	11
who_enum_str	Enumerated String	6
when_enum_str	Enumerated String	5
fail_enum_str	Enumerated String	10
request_enum_str	Enumerated String	2

report_enum_str	Enumerated String	7
recon_enum_str	Enumerated String	7
nbc_desc_enum_str	Enumerated String	10
burst_enum_str	Enumerated String	6
delivery_enum_str	Enumerated String	7
cloud_hgt_enum_str	Enumerated String	4
description_enum_str	Enumerated String	8
type_enum_str	Enumerated String	6
visibility_enum_str	Enumerated String	8
restriction_enum_str	Enumerated String	10
cloud_enum_str	Enumerated String	5
turbint_enum_str	Enumerated String	6
turbfreq_enum_str	Enumerated String	5
iceint_enum_str	Enumerated String	6
icing_hgt_enum_str	Enumerated String	4
aircraft_enum_str	Enumerated String	13
aircraft_status_enum_str	Enumerated String	5
pilot_status_enum_str	Enumerated String	6

#### 5.1.11 decode\_list\_proto.h

This header file prototypes the string arrays defined in `decode_defines.h`, section 5.1.6, and enumerated in `decode_strs.h`, section 5.1.10, for use by the decode modules.

#### 5.1.12 decode\_vars.h

This header file provides the data structures and definitions needed for global commonality among the modules that make up the decode CSC of the DMCC. This header includes the following header files; `DMC_PDU.h`, `decode_defines.h`, `decode_strs_proto.h`, `decode_list_proto.h`, and `decode_fprotos.h`. These header files, included in this order, dimension and define the globally used structures and data entities.

#### 5.1.13 DIS\_PDU.h

This header contains the definitions of the PDU's which define the DIS 1.0 protocol, including the PDU's which are part of the protocol extensions. This header file implements the requirement for entity information and entity interaction protocol data units (PDU's) exchanged between simulators participating in a distributed interactive simulation.

5.1.13.1.1 DIS\_PDU structure Header\_type

Table 22 DIS\_PDU structure Header\_type

Item Name	Type	bytes	Description
version	Char	1	
exercise	Char	1	
type	Char	1	
unused	Char	1	

5.1.13.1.2 DIS\_PDU structure ActivateRequest\_type

Table 23 DIS\_PDU structure ActivateRequest\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
ActivateHostID	Struct		ID_type
NewEntityID	Struct		ID_type
DateTime	Long	4	
Reason	Char	1	
Padding_3	Char array	3	
TerrainDBaseID	Struct		TerrainDBaseID_type
EntityType	Struct		Entity_type
Guise	Struct		Entity_type
Unit	Struct		Unit_type
Marking	Struct		EntityMarking_type
EntityCapabilities	Struct		EntityCapabilities_type
SubsystemParameters	Double	8	
EntityLocation	Struct		WorldCoord_type
Orientation	Struct		Orientation_type
DefaultUpdateThresholds	Struct		DefaultThresholds_type
NumberOfRadarSystems	Char	1	
NumberOfStores	Char	1	
NumberOfArticulationParameters	Char	1	
Padding	Char	1	
RadarSystems	Array of Struct		RadarSystems_type
Stores	Array of Struct		Supply_type
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.3 DIS\_PDU structure ActivateResponse\_type

Table 24 DIS\_PDU structure ActivateResponse\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
Timelimit	Integer	2	
ActivateResult	Char	1	
Padding_3	Char array	3	

5.1.13.1.4 DIS\_PDU structure DeactivateRequest\_type

Table 25 DIS\_PDU structure DeactivateRequest\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
DeactivateReason	Char	1	
Padding	Char	1	

5.1.13.1.4 DIS\_PDU structure DeactivateResponse\_type

Table 26 DIS\_PDU structure DeactivateResponse\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
DeactivateResult	Char	1	
Padding	Char	1	



5.1.13.1.5 DIS\_PDU structure Collision\_type

Table 27 DIS\_PDU structure Collision\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
IssuingEntityID	Struct		ID_type
CollidingEntityID	Struct		ID_type
EventID	Struct		Event_ID_type
Padding_2	Integer	2	
TimeStamp	Struct		TimeStamp_type
Velocity	Struct		LinearVelocity_type
Mass	Float	4	
Location	Struct		EntityCoord_type

5.1.13.1.6 DIS\_PDU structure Detonation\_type

Table 28 DIS\_PDU structure Collision\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
FiringEntityID	Struct		ID_type
TargetEntityID	Struct		ID_type
MunitionID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Location	Struct		WorldCoord_type
BurstLocation	Struct		BurstLocation_type
Velocity	Struct		LinearVelocity_type
Impact	Struct		EntityCoord_type
DetonationResult	Char	1	
NumberOfArticulationParameters	Char	1	
Padding	Char array	6	
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.7 DIS\_PDU structure Emitter\_type

Table 29 DIS\_PDU structure Emitter\_Type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EmittingEntityID	Struct		ID_type
Padding_2	Integer	2	
EntityType	Struct		Entity_type
TimeStamp	Struct		TimeStamp_type
EntityLocation	Struct		WorldCoord_type
EntityLinearVelocity	Struct		LinearVelocity_type
EntityOrientation	Struct		Orientation_type
DeadReckoningParameters	Struct		DeadReckoning_type
NumberOfEmitters	Integer	2	
DatabaseNumber	Integer	2	
Emitter	Array of Struct		Emitters_type

5.1.13.1.8 DIS\_PDU structure EntityState\_type

Table 30 DIS\_PDU structure EntityState\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EntityID	Struct		ID_type
Padding	Char	1	
ForceID	Char	1	
EntityType	Struct		Entity_type
Guise	Struct		Entity_type
TimeStamp	Struct		TimeStamp_type
EntityLocation	Struct		WorldCoord_type
EntityLinearVelocity	Struct		LinearVelocity_type
EntityOrientation	Struct		Orientation_type
DeadReckoningParameters	Struct		DeadReckoning_type
EntityAppearance	Struct		EntityAppearance_type
EntityMarking	Struct		EntityMarking_type
Capabilities	Struct		EntityCapabilities_type
Padding_3	Char array	3	
NumberOfArticulationParameters	Char	1	
ArticulationParameters	Array of Struct		Articulation_type

5.1.13.1.9 DIS\_PDU structure Fire\_typeTable 31 DIS\_PDU structure Fire\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
FiringEntityID	Struct		ID_type
TargetEntityID	Struct		ID_type
MunitionID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Location	Struct		WorldCoord_type
BurstDescription	Struct		BurstDescription_type
Velocity	Struct		LinearVelocity_type
Range	Float	4	

5.1.13.1.10 DIS\_PDU structure Radar\_typeTable 32 DIS\_PDU structure Radar\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
EmittingEntityID	Struct		ID_type
EventID	Struct		Event_ID_type
TimeStamp	Struct		TimeStamp_type
Padding_3	Char array	3	
NumberOfRadarSystems	Char	1	
RadarSystems	Array of Struct		RadarArray_type

5.1.13.1.11 DIS\_PDU structure RepairComplete\_typeTable 33 DIS\_PDU RepairComplete\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
Repair	Integer	2	
Padding_2	Integer	2	

5.1.13.1.12 DIS\_PDU structure RepairResponse\_type

Table 34 DIS\_PDU structure RepairResponse\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
RepairResult	Char	1	
Padding_3	Char array	3	

5.1.13.1.13 DIS\_PDU structure Resupply\_type

Table 35 DIS\_PDU structure Resupply\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
NumberOfSupplyTypes	Char	1	
Padding_3	Char array	3	
Supply	Array of Struct		Supply_type

5.1.13.1.14 DIS\_PDU structure ResupplyCancel\_type

Table 36 DIS\_PDU structure ResupplyCancel\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type

5.1.13.1.15 DIS\_PDU structure ServiceRequest\_type

Table 37 DIS\_PDU structure ServiceRequest\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestingEntityID	Struct		ID_type
ServicingEntityID	Struct		ID_type
ServiceType	Char	1	
NumberOfSupplyTypes	Char	1	
Padding_2	Integer	2	
Supply	Array of Struct		Supply_type

5.1.13.1.16 DIS\_PDU structure UpdateThresholdRequest\_type

Table 38 DIS\_PDU structure UpdateThresholdRequest\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestorEntityID	Struct		ID_type
ResponderEntityID	Struct		ID_type
LinearThreshold	Struct		EntityCoord_type
RotationalThreshold	Struct		Orientation_type
DurationOfChange	Integer	4	

5.1.13.1.17 DIS\_PDU structure UpdateThresholdResponse\_type

Table 39 DIS\_PDU structure UpdateThresholdResponse\_type

Item Name	Type	bytes	Description
Header	Struct	4	Header_type - 5.1.13.1.1
RequestorEntityID	Struct		ID_type
ResponderEntityID	Struct		ID_type
Result	Char	1	
RemainingTime	Char	1	
Padding_2	Integer	2	

## 5.1.14 DIS\_types.h

This header file contains definitions of various basic data elements, including how these elements are represented as communicated bits. The elements compose the fields which compose each PDU.

5.1.14.1.1 DIS\_types structure LandTable 40 DIS\_types structure Land

Item Name	Type	bytes	Description
Unused	Long	4	
Hatch	Long	4	
Engine_smoke	Long	4	
Launcher	Long	4	
Paint_scheme	Long	4	
Dust_cloud	Long	4	
Flaming	Long	4	
Smoke_plume	Long	4	
Destroyed	Long	4	

5.1.14.1.2 DIS\_types structure AirTable 41 DIS\_types structure Land

Item Name	Type	bytes	Description
Unused	Long	4	
Formation_lights	Long	4	
Navigation_lights	Long	4	
Running_lights	Long	4	
After_burner	Long	4	
Flaming	Long	4	
Destroyed	Long	4	

5.1.14.1.3 DIS\_types structure SurfaceTable 42 DIS\_types structure Surface

Item Name	Type	bytes	Description
Unused	Long	4	
Running_lights	Long	4	
Wake	Long	4	
Flaming	Long	4	
Destroyed	Long	4	

5.1.14.1.4 DIS\_types structure SubsurfaceTable 43 DIS\_types structure Subsurface

Item Name	Type	bytes	Description
Unused	Long	4	
Hatch	Long	4	
Running_lights	Long	4	
Wake	Long	4	
Destroyed	Long	4	

5.1.14.1.5 DIS\_types structure SpaceTable 44 DIS\_types structure Space

Item Name	Type	bytes	Description
Unused	Long	4	
Destroyed	Long	4	

5.1.14.1.6 DIS\_types structure GuidedMunitionsTable 45 DIS\_types structure GuidedMunitions

Item Name	Type	bytes	Description
Unused	Long	4	
Rocket_flame	Long	4	
Launch_flash	Long	4	

5.1.14.1.7 DIS\_types structure LifeformsTable 46 DIS\_types structure Lifeforms

Item Name	Type	bytes	Description
Unused	Long	4	
Weapon_1	Long	4	
Weapon_2	Long	4	
Lifeform_state	Long	4	

5.1.14.1.8 DIS\_types structure EnvironmentalsTable 47 DIS\_types structure Environmentals

Item Name	Type	bytes	Description
Unused	Long	4	
Damage	Long	4	

5.1.14.1.9 DIS\_types structure CulturalFeaturesTable 48 DIS\_type structure CulturalFeatures

Item Name	Type	bytes	Description
Unused	Long	4	
Flaming	Long	4	
Smoke_plume	Long	4	
Damage	Long	4	



5.1.14.1.10 DIS\_types structure ID\_typeTable 49 DIS\_types structure ID\_type

Item Name	Type	bytes	Description
Site	Integer	2	
Host	Integer	2	
Entity	Integer	2	

5.1.14.1.11 DIS\_types structure Event\_ID\_typeTable 50 DIS\_types structure Event\_ID\_type

Item Name	Type	bytes	Description
Site	Integer	2	
Host	Integer	2	
Event	Integer	2	

5.1.14.1.12 DIS\_types structure Entity\_typeTable 51 DIS\_types structure Event\_ID\_type

Item Name	Type	bytes	Description
EntityKind	Char	1	
Domain	Char	1	
Country	Integer	2	
Category	Char	1	
Subcategory	Char	1	
specific	Char	1	
Extra	Char	1	

5.1.14.1.13 DIS\_types structure Supply typeTable 52 DIS\_types structure Supply type

Item Name	Type	bytes	Description
Entity	Struct		Entity_type - 5.1.13.1.18
Quantity	Float	4	

5.1.14.1.15 DIS\_types structure DefaultThresholds typeTable 53 DIS\_types structure DefaultThresholds type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	
Psi	Long	4	
Theta	Long	4	
Phi	Long	4	

5.1.14.1.16 DIS\_types structure WorldCoord typeTable 54 DIS\_types structure WorldCoord type

Item Name	Type	bytes	Description
X	Double	8	
Y	Double	8	
Z	Double	8	

5.1.14.1.17 DIS\_types structure EntityCoord typeTable 55 DIS\_types structure EntityCoord type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	

5.1.14.1.18 DIS\_types structure AngularVelocity\_typeTable 56 DIS\_types structure AngularVelocity\_type

Item Name	Type	bytes	Description
Psi	Integer	4	
Theta	Integer	4	
Phi	Integer	4	

5.1.14.1.19 DIS\_types structure FloatAngles\_typeTable 57 DIS\_types structure FloatAngles\_type

Item Name	Type	bytes	Description
Psi	Float	4	
Theta	Float	4	
Phi	Float	4	

5.1.14.1.20 DIS\_types structure LinearVelocity\_typeTable 58 DIS\_types structure LinearVelocity\_type

Item Name	Type	bytes	Description
X	Float	4	
Y	Float	4	
Z	Float	4	

5.1.14.1.21 DIS\_types structure Orientation\_typeTable 59 DIS\_types structure Orientation\_type

Item Name	Type	bytes	Description
Psi	Long	4	
Theta	Long	4	
Phi	Long	4	

5.1.14.1.22 DIS\_types structure DeadReckoning\_typeTable 60 DIS\_types structure DeadReckoning\_type

Item Name	Type	bytes	Description
DeadReckoningAlgorithm	Char	1	
OtherParameters	Char array	15	
LinearAcceleration	Struct	12	LinearVelocity_type - 5.1.14.1.20
AngularVelocity	Struct	12	AngularVelocity_type - 5.1.14.1.18

5.1.14.1.23 DIS\_types structure EntityMarking\_typeTable 61 DIS\_types structure EntityMarking\_type

Item Name	Type	bytes	Description
CharacterSet	Char	1	
CharacterString	Char array	11	

5.1.14.1.24 DIS\_types structure TerrainDBaseID\_typeTable 62 DIS\_types structure TerrainDBaseID\_type

Item Name	Type	bytes	Description
CharacterString	Char array	11	
VersionNumber	Char	1	

5.1.14.1.25 DIS\_types structure BurstDescription typeTable 63 DIS\_types structure BurstDescription type

Item Name	Type	bytes	Description
Munition	Struct	6	Entity_type
Warhead	Integer	2	
Fuze	Integer	2	
Quantity	Integer	2	
Rate	Integer	2	

5.1.14.1.26 DIS\_types structure ArmyTable 64 DIS\_types structure Army

Item Name	Type	bytes	Description
Army_Number	Char	1	
Corps_Number	Char	1	
Division_Number	Char	1	
Regiment_Number	Char	1	
Battalion_Number	Char	1	
Company_Number	Char	1	
Platoon_Number	Char	1	
Squad_Number	Char	1	

5.1.14.1.27 DIS\_types structure AirForceTable 65 DIS\_types structure AirForce

Item Name	Type	bytes	Description
AirForce_Number	Char	1	
Wing_Number	Char	1	
Squadron_Number	Char	1	
Flight_Number	Char	1	
Division_Number	Char	1	
Section_Number	Char	1	
Spare	Char array	2	

5.1.14.1.28 DIS\_types structure CoastGuardTable 66 DIS\_types structure CoastGuard

Item Name	Type	bytes	Description
Spare	Char array	8	

5.1.14.1.29 DIS\_types structure MarinesTable 67 DIS\_types structure Marines

Item Name	Type	bytes	Description
Fleet_Number	Char	1	
Force_Number	Char	1	
Brigade_Number	Char	1	
Unit_Number	Char	1	
Spare	Char array	4	

5.1.14.1.30 DIS\_types structure NavyTable 68 DIS\_types structure Navy

Item Name	Type	bytes	Description
Fleet_Number	Char	1	
Force_Number	Char	1	
Group_Number	Char	1	
Unit_Number	Char	1	
Element_Number	Char	1	
Spare	Char array	3	

5.1.14.1.31 DIS\_types structure Unit\_typeTable 69 DIS\_types structure Unit\_type

Item Name	Type	bytes	Description
Force	Char	1	
Country	Char array	2	
Service	Char	1	
Hierarchy	Union of Struct	8	Either Army, AirForce, CoastGuard, Marines, Navy - 5.1.14.1.26 to 5.1.14.1.30

5.1.14.1.32 DIS\_types structure EntityCapabilities\_typeTable 70 DIS\_types structure EntityCapabilities\_type

Item Name	Type	bytes	Description
Unused	Long	4	
Repair	Long	4	
MiscellaneousSupply	Long	4	
FuelSupply	Long	4	
AmmunitionSupply	Long	4	

5.1.14.1.33 DIS\_types structure RadarSystems\_typeTable 71 DIS\_types structure RadarSystems\_type

Item Name	Type	bytes	Description
Category	Char	1	
Subcategory	Char	1	
RadarSystemID	Integer	2	

5.1.14.1.34 DIS\_types structure Illumined\_typeTable 72 DIS\_types structure Illumined\_type

Item Name	Type	bytes	Description
TargetID	Struct		ID_type
Padding_2	Integer	2	
RadarData	Long	4	

5.1.14.1.35 DIS\_types structure Sweep\_typeTable 73 DIS\_types structure Sweep\_type

Item Name	Type	bytes	Description
AzimuthCenter	Long	4	
AzimuthWidth	Long	4	
ElevationCenter	Long	4	
ElevationWidth	Long	4	

5.1.14.1.36 DIS\_types structure Emitters\_Type\_typeTable 74 DIS\_types structure Emitters\_Type\_type

Item Name	Type	bytes	Description
Class	Char	1	
ModeNumber	Char	1	
DBaseEntryNumber	Integer	2	

5.1.14.1.37 DIS\_types structure Emitters\_Data\_typeTable 75 DIS\_types structure Emitters\_Data\_type

Item Name	Type	bytes	Description
Power	Integer	2	
ParameterNumber1	Integer	2	
ParameterNumber2	Integer	2	
ParameterNumber3	Integer	2	



5.1.14.1.38 DIS\_types structure Emitters\_typeTable 76 DIS\_types structure Emitters\_type

Item Name	Type	bytes	Description
Type	Struct	4	Emitters_Type_type - 5.1.14.1.36
Parameters	Struct	8	Emitters_Data_type - 5.1.14.1.37
Sweep	Struct	16	Sweep_type - 5.1.14.1.35

5.1.14.1.39 DIS\_types structure RadarBaseDate\_typeTable 77 DIS\_types structure RadarBaseDate\_type

Item Name	Type	bytes	Description
Location	Struct		EntityCoord_type
RadarSystem	Struct	8	RadarSystems_type
Power	Integer	2	
RadarMode	Char	1	
NumberOfIlluminated	Char	1	
SpecificData	Long array	8	
Sweep	Struct	16	Sweep_type - 5.1.14.1.35

5.1.14.1.40 DIS\_types structure RadarArray\_typeTable 78 DIS\_types structure RadarArray\_type

Item Name	Type	bytes	Description
RadarSystemData	Struct	4	RadarBaseData_type
Illumined	Array of Struct		Illumined_type

5.1.14.1.41 DIS\_types structure TimeStamp typeTable 79 DIS\_types structure TimeStamp type

Item Name	Type	bytes	Description
value	Long	4	
reference	Long	4	

## 5.1.14.2 DIS\_types define statements

Table 80 DIS\_types define statements

Name	Value	Description
LIFEFORM STATE DESTROYED	0	
LIFEFORM STATE UPRIGHT	1	
LIFEFORM STATE KNEELING	2	
LIFEFORM STATE PRONE	3	
PASS ALL ID CODE	0xFFFF	
IGNORE ALL ID CODE	0	
ENTITY ID SIZE	6	
OTHER KIND	0	
PLATFORM KIND	1	
MUNITION KIND	2	
LIFE FORM KIND	3	
ENVIRONMENTAL KIND	4	
CULTURAL FEATURE KIND	5	
SUPPLIES KIND	6	
LAND DOMAIN	1	
AIR DOMAIN	2	
SURFACE DOMAIN	3	
SUBSURFACE DOMAIN	4	
SPACE DOMAIN	5	
TANK	1	
ARMORED FIGHTING VEH	2	
ARMORED UTILITY VEH	3	
SELF PROPELLED ARTILLERY	4	
TOWED ARTILLERY	5	
SMALL WHEELED UTILITY VEH	6	
LARGE WHEELED UTILITY VEH	7	
SMALL TRACKED UTILITY VEH	8	
LARGE TRACKED UTILITY VEH	9	
DIS MORTAR	10	
AMPHIBIOUS ASSAULT VEH	50	
MISC WEAPONS SYSTEMS	51	
DIS M1 TANK	1	

DIS M60 TANK	2	
DIS M551 TANK	3	
DIS M48 TANK	4	
DIS IFV AFV	1	
DIS M2 AFV	3	
DIS M3 AFV	4	
DIS M113 AFV	5	
DIS HUMMV SWUV	1	
DIS M577 SWUV	2	
DIS FAV SWUV	3	
DIS M352 LWUV	1	
DIS M977 LWUV	2	
DIS M978 LWUV	3	
DIS M548 LTUV	1	
DIS MLRS SPA	1	
DIS M110 SPA	2	
DIS M109 SPA	3	
DIS GLCM LAUNCHER SPA	4	
DIS M198 155MM TOWED ART	2	
DIS M57 TOWED ART	11	
DIS M128 GEMSS TOWED ART	12	
DIS M58A1 TOWED ART	14	
DIS M88 AUV	1	
DIS M578 AUV	2	
DIS M29 MORTAR	1	
DIS M224 MORTAR	2	
DIS MK19 GRENADE LAUNCHER	2	
DIS VULCAN ADS SAM	3	
DIS M113A2 MODEL	1	
DIS M88A1 MODEL	1	
DIS M352 MODEL	1	
DIS M352A2 MODEL	2	
DIS USSR T80 TANK	1	
DIS USSR T72M TANK	2	
DIS USSR T64 TANK	3	
DIS USSR T62 TANK	4	
DIS USSR T54 TANK	5	
DIS USSR T55 TANK	6	
DIS USSR T10 TANK	7	
DIS USSR PT76 TANK	8	
DIS USSR D20 TOWED ART	3	
DIS USSR D30 122MM TOWED ART	4	
DIS USSR D30FIELD TOWED ART	6	
DIS USSR M1965 TOWED ART	10	
DIS USSR M1943 MORTAR	2	
DIS USSR M1952 MORTAR	4	
DIS USSR SS12 SPA	11	
DIS USSR BM21 MRLS SPA	13	
DIS USSR BM24 MRLS SPA	15	

DIS USSR BMP1 AFV	1	
DIS USSR BMP2 AFV	2	
DIS USSR BRDM1 AFV	3	
DIS USSR BRDM2 AFV	4	
DIS USSR BTR40 AFV	5	
DIS USSR BTR153 AFV	6	
DIS USSR BTR60 70 AFV	8	
DIS USSR ACRV AFV	10	
DIS USSR BTS2 AUV	1	
DIS USSR MTP AUV	2	
DIS USSR IMR2 AUV	3	
DIS USSR IPR AUV	4	
DIS USSR MAV SWUV	3	
DIS USSR GAZ63 LWUV	7	
DIS USSR GAZ66 LWUV	8	
DIS USSR URAL375 LWUV	9	
DIS USSR FROG5 SSML	8	
DIS USSR FROG7 SSML	9	
DIS USSR SCUDB SSML	10	
DIS USSR PTS XPORT LTUV	1	
USSR BMP1K AFV MODEL	1	
USSR T72 TANK MODEL	1	
USSR T72M TANK MODEL	2	
FIGHTER	1	
ATTACK	2	
DIS BOMBER	3	
CARGO	4	
ANTI SUBMARINE	5	
PATOL	5	
ATTACK HELICOPTER	6	
UTILITY HELICOPTER	7	
CARGO HELICOPTER	50	
OBSERVATION HELICOPTER	51	
SEA GOING HELICOPTER	52	
ELECTRONIC WARFARE	53	
DIS A4 ATTACK	1	
DIS A7 ATTACK	3	
DIS A10 ATTACK	4	
DIS F117 STEALTH	1	
DIS F14 FIGHTER	2	
DIS F16 FIGHTER	3	
DIS FB 111 FIGHTER	4	
DIS F4S FIGHTER	5	
DIS F15 FIGHTER	7	
DIS FA 18 FIGHTER	14	
DIS F5F FIGHTER	16	
DIS B52 BOMBER	3	
DIS C130 CARGO	1	
DIS C5 CARGO	2	

CIS C141 CARGO	3	
DIS P3 PATROL	1	
DIS PIONEER RPV PATROL	8	
DIS E2 ELEC WARFARE	1	
DIS MODEL F14A	1	
DIS MODEL F14D	4	
DIS AH64 COPTER	1	
DIS AH1 COPTER	2	
DIS OH58 COPTER	1	
DIS OH58C COPTER	2	
DIS OH58D COPTER	3	
DIS UH1 COPTER	1	
DIS UH60 COPTER	2	
DIS CH47 COPTER	1	
DIS CH53 COPTER	2	
DIS MIG31 FIGHTER	1	
DIS MIG29 FIGHTER	2	
DIS SU27 FIGHTER	3	
DIS MIG25 FIGHTER	4	
DIS MIG23 FIGHTER	5	
DIS MIG21 FIGHTER	6	
DIS MIG27 ATTACK	1	
DIS SU25 ATTACK	8	
DIS YAK36 ATTACK	10	
DIS TU16 BOMBER	2	
DIS TU26 BOMBER	3	
DIS TU22 BOMBER	4	
DIS TU142 BOMBER	5	
DIS TU126 PATROL	3	
DIS AN124 CARGO	1	
DIS AN22 CARGO	2	
DIS II76 CARGO	3	
DIS MI28 COPTER	1	
DIS MI24 COPTER	2	
DIS KA136 COPTER	3	
DIS MI8 COPTER	1	
DIS MI14 COPTER	2	
DIS MI26 COPTER	3	
DIS MI6 COPTER	4	
DIS MI2 COPTER	5	
DOS MIRAGE F1 FIGHTER	1	
DIS SA342 GAZELLE	5	
DIS UK CHIEFTAIN TANK	1	
CARRIER	1	
COMMAND SHIP	2	
GUIDED MISSILE CRUISER	3	
GUIDED MISSILE DESTROYER	4	
DESTROYER	5	
GUIDED MISSILE FRIGATE	6	

PATROL CRAFT	7	
MINE COUNTERMEASURE	8	
DOCK LANDING SHIP	9	
TANK LANDING SHIP	10	
LANDING CRAFT	11	
FRIGATE	50	
BATTLESHIP	51	
HEAVY CRUISER	52	
DESTROYER TENDER	53	
AMPHIBIOUS ASSAULT SHIP	54	
AMPHIBIOUS CARGO SHIP	55	
AMPHIBIOUS TRANSPORT DOCK	56	
AMMUNITION SHIP	57	
COMBAT STORES SHIP	58	
DIS USS TAWARA CLASS	1	
DIS USS IWO JIMA CLASS	2	
DIS USS WASP CLASS	3	
DIS USS CHARLESTON CLASS	1	
DIS USS AUSTIN CLASS	1	
DIS USS RALIEGH CLASS	2	
DIS USS WHIDBEY ISLANE CLASS	1	
DIS USS ANCHORAGE CLASS	2	
DIS USS THOMASTON CLASS	3	
DIS USS NEWPORT CLASS	1	
DIS USS DESOTO COUNTY CLASS	2	
DIS USS LCAC CLASS	1	
DIS USS NIMITZ CLASS	1	
DIS USS ENTERPRISE CLASS	2	
DIS USS KITTYHAWK CLASS	3	
DIS USS FORRESTAL CLASS	4	
DIS USS MIDWAY CLASS	5	
DIS USS BLUERIDGE CLASS	1	
DIS MISC CONVERT CMD SHIPS	2	
DIS USS TICONDEROGA CLASS	1	
DIS USS VIRGINIA CLASS	2	
DIS USS CALIFORNIA CLASS	3	
DIS USS TRUXTUN CLASS	4	
DIS USS BELKNAP CLASS	5	
DIS USS BAINBRIDGE CLASS	6	
DIS USS LEAHY CLASS	7	
DIS USS ALBANY CLASS	8	
DIS USS LONGBEACH CLASS	9	
DIS USS ARLEIGHBURKE CLASS	1	
DIS USS KIDD CLASS	2	
DIS USS FARRAGUT CLASS	3	
DIS USS CHARLESFADAMS CLASS	4	
DIS USS SPRUANCE CLASS	1	
DIS USS OHPERRY CLASS	1	
DIS USS BROOKE CLASS	2	

DIS USS PEGASUS CLASS	1	
DIS USS HIGHPOINT CLASS	2	
DIS USS MM SPEC WARFARE CLASS	3	
DIS USS KNOX CLASS	1	
DIS USS GARCIA CLASS	2	
DIS USS BFONSTEIN CLASS	3	
DIS USS IOWA CLASS	1	
DIS USS DESMOINES CLASS	1	
DIS USS SAM GOMPERS CLASS	1	
DIS USS DIXIE CLASS	2	
DIS USS KILAUEA CLASS	1	
DIS USS MARS CLASS	2	
DIS DD 964 PAULFFOSTER	1	
DIS DD 963 SPRUANCE	2	
DIS DD 976 MERRILL	14	
DIS DD 978 STUMP	16	
DIS DDG 37 FARRAGUT	1	
DIS DDG 39 MACDONOUGH	3	
DIS DDG 40 COONTZ	4	
DIS CG47 TICONDEROGA	1	
DIS CG48 YORKTOWN	2	
DIS CG10 ALBANY	1	
DIS PCH 1	1	
DIS PBM 1	1	
DIS CA134 DESMOINES	1	
DIS FF1037 BRONSTEIN	1	
DIS LCAC	1	
DIS TAE26 KILAUEA	1	
DIS TAFS8 SIRIUS	1	
DIS AFS1 MARS	1	
DIS AD41 YELLOWSTONE	3	
DIS AD42 ACADIA	4	
DIS AD15 PRAIRIE	1	
MCM MSH TYPE	1	
MCM AVENGER CLASS	2	
MCM ACME CLASS	3	
MCM AGILE AGRESS CLASS	4	
MSM 1 AVENGER	1	
MSM 509 ADROIT	1	
MSM 427 CONSTANT	1	
STRATEGIC MISSILE SUB	1	
ATTACK SUB	2	
RESEARCH SUB	50	
DIS USS OHIO CLASS	1	
DIS USS BEN FRANKLIN CLASS	2	
DIS USS LOS ANGELES CLASS	1	
DIS USS STURGEON CLASS	4	
DIS USS SKIPJACK CLASS	7	
DIS USS BARBEL CLASS	8	

DIS USS TRITON CLASS	10	
FRANKLIN CLASS OFFSET	19	
DIS USS DOLPHIN CLASS	1	
DIS SSBN 616 LAFAYETTE	1	
DIS SSN 585 SKIPJACK	1	
DIS SSN 590 SCULPIN	3	
DIS SSN 590 SNOOK	5	
DIS SSN 586 TRITON	1	
DIS AGSS 555 DOLPHIN	1	
MANNED SPACE PLATFORM	1	
UNMANNED SPACE PLATFORM	2	
SATELLITE USP	1	
SPACE SHUTTLE MSP	1	
SPACE STATION MSP	2	
GUIDED MUNITIONS	1	
BALLISTIC MUNITIONS	2	
FIXED MUNITIONS	3	
BOMB MUNITIONS	4	
FUZE MUNITIONS	5	
DIS TMD B	1	
DIS YAM 5	2	
DIS TM 46	5	
DIS TMN 46	6	
DIS TM 57	7	
DIS TM 62M	8	
DIS TMK 2	9	
DIS POMZ 2	10	
DIS OMZ 3	11	
DIS MON	12	
DIS PMN	13	
DIS PMK 40	14	
ANTI AIR DOMAIN	1	
ANTI ARMOR DOMAIN	2	
ANTI GUIDED MUNITION DOMAIN	3	
ANTI RADAR DOMAIN	4	
ANTI SATELLITE DOMAIN	5	
ANTI SHIP DOMAIN	6	
ANTI SUBMARINE DOMAIN	7	
BATTLEFIELD SUPPORT DOMAIN	8	
STRATEGIC DOMAIN	9	
MISCELLANEOUS MUNITIONS DOMAIN	10	
ANTI PERSONNEL	1	
ANTI ARMOR	2	
BATTLEFIELD SUPPORT	3	
DIS SIDEWINDER	1	
DIS ADATS	2	
DIS STINGER	3	
TOMAHAWK AGM 109	5	
TOMAHAWK BGM 109G	6	



DIS GREMLIN	1	
TOW MISSILE	1	
M47 DRAGON	2	
DIS SWATTER	4	
DIS SPANDREL	7	
DIS SPIRAL	8	
DIS SONGSTER	10	
DIS STYX	8	
LAND LIFEFORM	1	
DIS OTHER INFANTRY	0	
DIS DISMOUNTED INFANTRY	1	
NUM TEAM	101	
NUM SQUAD	102	
NUM PLATOON	103	
NUM COMPANY	104	
NUM BATTALION	105	
NUM REGIMENT	106	
NUM DIVISION	107	
NUM CORPS	108	
NUM ARMY	109	
DIS M72 LAW	4	
DIS M60 GPMG	6	
DIS RPG7 VAT ROCKET LAUNCHER	4	
DIS AA52 MG	6	
BRIDGE CONCRETE TWO LANE	1	
BRIDGE CONCRETE FOUR LANE	2	
BRIDGE TRUSS TWO LANE	3	
BRIDGE TRUSS FOUR LANE	4	
BRIDGE SUSPENSION TWO LANE	5	
BRIDGE SUSPENSION FOUR LANE	6	
BUILDING ONE STORY	7	
BUILDING TWO STORY	8	
BUILDING THREE STORY	9	
BUILDING FOUR STORY	10	
MINEFIELD MARKERS	17	
BREACHED LANE FLAGS	18	
SIZE VERY SMALL	20	
SIZE SMALL	40	
SIZE MEDIUM	60	
SIZE LARGE	80	
SIZE VERY LARGE	100	
FUELS	1	
OILS	2	
LUBRICANTS	3	
FOOD	4	
SPARE PARTS	5	
PERSONNEL	6	
DIS PROPELLANT	7	
DIS CONSTRUCTION	8	

DIS AMMO	9	
DIS PERSONAL DEMAND	9	
DIS MAJOR END ITEMS	10	
DIS MEDICAL MATERIAL	11	
GASOLINE	1	
DIESEL FUEL	2	
JP4 FULE	3	
FUEL OIL	4	
DIS BAGGED PROPELLANT	1	
DIS CANNISTERED PROPELLANT	2	
DIS AIR DEFENSE AMMO	1	
DIS AIR FORCE AMMO	2	
DIS ARMY AVIATION AMMO	3	
DIS ARTILLERY AMMO	4	
DIS CHEMICAL AMMO	5	
DIS MINES AND EXPLOSIVES	6	
DIS NUCLEAR AMMO	7	
DIS ROCKET ARTILLERY AMMO	8	
DIS SMALL ARMS AMMO	9	
DIS TANK AMMO	10	
DIS COUNTRY OTHER	0	
DIS COUNTRY FRANCE	55	
DIS COUNTRY GERMANY	57	
DIS COUNTRY USSR	164	
DIS COUNTRY UK	166	
DIS COUNTRY USA	168	
GUISE SIZE	8	
SUPPLY SIZE	12	
DRM OTHER	0	
DRM STATIC	1	
DRM FPW	2	
DRM RPW	3	
DRM RVW	4	
DRM FVW	5	
DRM FPB	6	
DRM RPB	7	
DRM RVB	8	
DRM FVB	9	
UNUSED DR BYTES	15	
ARTICULATION SIZE	16	
NO ART CHANGE	0	
DIRECTLY ATTACHED	0	
INDIRECTLY ATTACHED	1	
PARAM TYPE EMPTY	0	
PARAM TYPE POSITION	1	
PARAM TYPE POSITION RATE	2	
PARAM TYPE EXTENSION	3	
PARAM TYPE EXTENSION RATE	4	
PARAM TYPE LOCATION X	5	

PARAM TYPE LOCATION X RATE	6	
PARAM TYPE LOCATION Y	7	
PARAM TYPE LOCATION Y RATE	8	
PARAM TYPE LOCATION Z	9	
PARAM TYPE LOCATION Z RATE	10	
PARAM TYPE AZIMUTH	11	
PARAM TYPE AZIMUTH RATE	12	
PARAM TYPE ELEVATION	13	
PARAM TYPE ELEVATION RATE	14	
PARAM TYPE ROTATION	15	
PARAM TYPE ROTATION RATE	16	
PARAM TYPE ACCELERATION X	17	
PARAM TYPE ACCELERATION Y	18	
PARAM TYPE ACCELERATION Z	19	
NUM PARAM TYPES	20	
STATION RUDDER	1000	
STATION LEFT WING	1020	
STATION RIGHT WING	1040	
STATION LEFT AILERON	1060	
STATION RIGHT AILERON	1080	
STATION PERISCOPE	2000	
STATION ANTENNA	2020	
STATION SNORKEL	2040	
STATION LANDING GEAR	3000	
STATION TAIL HOOK	3020	
STATION SPEED BRAKE	3040	
STATION LEFT WEAPON BAY DOOR	3060	
STATION RIGHT WEAPON BAY DOOR	3080	
STATION HATCH	3100	
STATION WINGSWEEP	3120	
STATION PRIMARY TURRET	4000	
STATION PRIMARY GUN	4200	
STATION PRIMARY LAUNCHER	4400	
STATION PRIMARY DEFENSE	4600	
STATION PRIMARY RADAR	4800	
STATION SECONDARY TURRET	5000	
STATION SECONDARY GUN	5200	
STATION SECONDARY LAUNCHER	5400	
STATION SECONDARY DEFENSE	5600	
STATION SECONDARY RADAR	5800	
MAX ENTITY MARKING LENGTH	11	
CHARACTER SET UNUSED	0	
CHARACTER SET ASCII	1	
MAX DBASE ID LENGTH	11	
WARHD OTHER	0	
WARHD HE	1000	
WARHD HE PLASTIC	1100	
WARHD HE INCENDIARY	1200	
WARHD HE FRAGMENTAION	1300	

WARHD HE ANTI TANK	1400	
WARHD HE BOMBLETS	1500	
WARHD HE SHAPED CHARGE	1600	
WARHD SMOKE	2000	
WARHD ILLUMINATION	3000	
WARHD PRACTICE	4000	
WARHD KINETIC	5000	
WARHD UNUSED	6000	
WARHD NUCLEAR	7000	
WARHD CHEMICAL	8000	
WARHD CHEMICAL BLISTER	8100	
WARHD CHEMICAL BLOOD	8200	
WARHD CHEMICAL NERVE	8300	
WARHD BIOLOGICAL	9000	
FUZE OTHER	0	
FUZE CONTACT	1000	
FUZE CONTACT INSTANT	1100	
FUZE CONTACT DELAYED	1200	
FUZE TIMED	2000	
FUZE PROXIMITY	3000	
FUZE COMMAND	4000	
FUZE ALTITUDE	5000	
FUZE DEPTH	6000	
FUZE ACOUSTIC	7000	
OTHER FORCES	0	
BLUE FORCES	1	
RED FORCES	2	
WHITE FORCES	3	
OTHER SERVICE	0	
ARMY SERVICE	1	
AIR FORCE SERVICE	2	
COAST GUARD SERVICE	3	
MARINES SERVICE	4	
NAVY SERVICE	5	
RADAR SYSTEM SIZE	4	
category shift	4	
RESERVED CATEGORY	1	
AIR BASED FIRE CONTROL	1	
AIR BASED SEARCH	1	
GROUND BASED FIRE CONTROL	1	
GROUND BASED SEARCH	0	
SEA BASED FIRE CONTROL	0	
SEA BASED SEARCH	0	
RADAR MODE OFF	0	
SEARCH	1	
DOPLER HPRF	2	
DOPPLER MPRF	3	
DOPPLER LPRF	4	
MONOPULSE	5	

ACQUISITION	6	
TRACKING	7	
TRACK WHILE SCAN	8	
TERRAIN FOLLOW	9	
DATA LINK	10	
MAX ILLUMINED	10	
RELATIVE TIME	0	
ABSOLUTE TIME	1	

### 5.1.15 DMC\_PDU.h

This header file contains definitions of the PDUs which define an extension of the DIS 1.0 protocol, called the Digital Message Communication.

#### 5.1.15.1.1 DIS\_types structure DMC\_CommonBlock\_type

Table 81 DIS\_types structure DMC\_CommonBlock\_type

Item Name	Type	bytes	Description
SenderID	Struct		ID_type
RadioID	Integer	2	
ExerciseTime	Integer	4	
Unused_15	Char array	15	SIM_UNUSED_ARRAY_SIZE
TimesForwarded	Char	1	
SenderTerminal	Char	1	
SnderPerson	Char	1	
Altitude	Integer	4	
TargetID	Struct		ID_type
TargetTerminal	Char	1	
TargetPerson	Char	1	
SenderCEOI	Char array	8	based on CEOI LENG
LastFwdCEOI	Char array	8	based on CEOI LENG
TargetCEOI	Char array	64	based on CEOI LENG and MAX CEOI RCVR
TransmitDTG	Struct		DTG_type
TransmitFrequency	Double	8	
TransmitLocation	Struct		UTM_type
FreeText	Char array	64	based on ANNOTATION
ZuluTime	Integer	2	
SerialNumber	Integer	2	
Priority	Char	1	
Overlay	Char	1	
Subtype	Char	1	
Variation	Char	1	

5.1.15.1.2 DIS\_types structure Header and CommonBlock typeTable 82 DIS\_types structure Header and commonBlock type

Item Name	Type	bytes	Description
version	Char	1	
exercise	Char	1	
type	Char	1	
unused	Char	1	
SenderID	Struct		ID type
RadioID	Integer	2	
TimeStamp	Struct		TimeStamp_type
EncodingScheme	Struct		EncodingScheme_type
Padding	Integer	2	
SampleRate	Integer	4	
Length	Integer	2	
Samples	Integer	2	
ProtocolID	Integer	4	
Unused 15	Char array	1	
TimesForwarded	Char	1	
SenderTerminal	Char	1	
SenderPerson	Char	1	
Altitude	Integer	4	
TargetID	Struct		ID_type
TargetTerminal	Char	1	
TargetPerson	Char	1	
SenderCEOI	Char array	8	based on CEOI LENG
LastFwdCEOI	Char array	8	based on CEOI LENG
TargetCEOI	Char array	64	based on CEOI LENG and MAX CEOI RCVR
TransmitDTG	Struct		DTG_type
TransmitFrequency	Double	8	
TransmitLocation	Struct		UTM_type
FreeText	Char array	64	based on ANNOTATION
ZuluTime	Integer	2	
SerialNumber	Integer	2	
Priority	Char	1	
Overlay	Char	1	
Subtype	Char	1	
Variation	Char	1	

5.1.15.1.3 DIS\_types structure ACK Specific typeTable 83 DIS\_types structure ACK Specific type

Item Name	Type	bytes	Description
OriginalSenderID	Struct		ID_type
OriginalSenderTerminal	Char	1	
OriginalSenderPerson	Char	1	
AcknowledgeID	Struct		ID_type
AcknowledgeTerminal	Char	1	
AcknowledgePerson	Char	1	
OriginalSenderCEOI	Char array	8	based on CEOI LENG
AcknowledgeCEOI	Char array	8	based on CEOI LENG
OriginalDTG	Struct		DTG_type

5.1.15.1.4 DIS\_types structure Ack typeTable 84 DIS\_types structure ACK type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC CommonBlock_type
Specific	Struct		Ack_Specific_type
AcknowledgeID	Struct		ID_type

5.1.15.1.5 DIS\_types structure DISAck typeTable 85 DIS\_types structure DISAck type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_and_CommonBlock_type
Specific	Struct		Ack_Specific_type

5.1.15.1.6 DIS\_types structure Spot Specific typeTable 86 DIS\_types structure Spot Specific type

Item Name	Type	bytes	Description
TimeSighted	Struct		DTG type
ObserverLocation	Struct		UTM type
TargetQuantity	Integer	2	
ObserverIntentions	Char	1	
TargetType	Char	1	
TargetActivity	Char	1	
TargetDirection	Char	1	
TargetSpeed	Struct		Speed type
TargetLocation	Struct		UTM type
TargetUnit	Char array	16	based on UNIT LENG
Unused_16	Char array	16	

5.1.15.1.7 DIS\_types structure Spot typeTable 87 DIS\_types structure Spot type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct		Spot Specific

5.1.15.1.8 DIS\_types structure DISSpot typeTable 88 DIS\_types structure DISSpot type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_and_CommonBlock type
Specific	Struct		Spot Specific type



5.1.15.1.9 DIS\_types structure BDA Specific typeTable 89 DIS\_types structure BDA Specific type

Item Name	Type	bytes	Description
StrikeTimeStart	Struct		DTG type
StrikeTimeEnd	Struct		DTG type
TargetCategory	Char	1	
TargetType	Char	1	
TargetsDestroyed	Integer	2	
PercentCoverage	Char	1	
Unused_3	Char array	3	

5.1.15.1.10 DIS\_types structure BDA typeTable 90 DIS\_types structure BDA type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct		BDA Specific type

5.1.15.1.11 DIS\_types structure DISBDA typeTable 91 DIS\_types structure DISBDA type

Item Name	Type	bytes	Description
Header_and_Common	Struct		Header_and_CommonBlock type
Specific	Struct		BDA Specific

5.1.15.1.12 DIS\_types structure GndRouteTable 92 DIS\_types structure GndRoute

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
ClassificationFormula	Char	1	
Unused 6	Char array	6	
ClassFormInfo	Char array	32	based on RTE ID LENG
RouteID	Char array	32	based on RTE ID LENG
Leftover	Char array	8	

5.1.15.1.13 DIS\_types structure AirRouteTable 93 DIS\_types structure AirRoute

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
Obstacles	Char	1	
Unused 6	Char array	6	
ClassFormInfo	Char array	32	based on RTE ID LENG
RouteID	Char array	32	based on RTE ID LENG
Leftover	Char array	8	

5.1.15.1.14 DIS\_types structure BridgeTable 94 DIS\_types structure Bridge

Item Name	Type	bytes	Description
Type	Char	1	
Damage	Char	1	
Spans	Char	1	
ConstrMaterial	Char	1	
Unused 4	Char array	4	
Length	Char array	4	based on KEYPAD 4
Width	Char array	4	based on KEYPAD 4
Height	Char array	4	based on KEYPAD 4
Under	Char array	4	based on KEYPAD 4
ConstrDescr	Char array	16	based on KEYPAD 16
ID	Char array	32	based on RTE ID LENG
SpanLength	Char array	4	based on KEYPAD 4
LoadClass	Char array	4	based on KEYPAD 4

5.1.15.1.15 DIS\_types structure LZ\_PZTable 95 DIS\_types structure LZ\_PZ

Item Name	Type	bytes	Description
ActivityLikely	Char	1	
Obstacles	Char	1	
Unused_6	Char array	6	
ObstaclesDescr	Char array	16	based on KEYPAD 16
ID	Char array	16	based on KEYPAD 16
LZ Size	Char array	16	based on KEYPAD 16
Axis	Char array	16	based on KEYPAD 16
Leftover	Char array	8	

5.1.15.1.16 DIS\_types structure BP\_OPTable 96 DIS\_types structure BP\_OP

Item Name	Type	bytes	Description
EnemyActivity	Char	1	
Obstacles	Char	1	
Unused_6	Char array	6	
ObstaclesDescr	Char array	16	based on KEYPAD 16
ID	Char array	16	based on KEYPAD 16
BP Size	Char array	16	based on KEYPAD 16
Axis	Char array	16	based on KEYPAD 16
Leftover	Char array	8	

5.1.15.1.17 DIS\_types structure CrossingTable 97 DIS\_types structure Crossing

Item Name	Type	bytes	Description
BankSlopeEntry	Char array	4	based on KEYPAD 4
BankSlopeExit	Char array	4	based on KEYPAD 4
CrossingLength	Char array	4	based on KEYPAD 4
CrossingWidth	Char array	4	based on KEYPAD 4
CrossingDepth	Char array	4	based on KEYPAD 4
CurrentFlow	Char array	4	based on KEYPAD 4
ID	Char array	8	based on KEYPAD 8
Leftover	Char array	48	

5.1.15.1.18 DIS\_types structure Recon Specific typeTable 98 DIS\_types structure Recon Specific type

Item Name	Type	bytes	Description
Subtype	Union		GndRoute, AirRoute, Bridge, LZ_PZ, BP_OP, Crossing

5.1.15.1.19 DIS\_types structure Recon typeTable 99 DIS\_types structure Recon type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct		Recon Specific type

5.1.15.1.20 DIS\_types structure DISRecon typeTable 100 DIS\_types structure DisRecon type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlo ck_type
Specific	Struct		Recon_Specific_type

5.1.15.1.21 DIS\_types structure RepeatTable 101 DIS\_types structure Repeat

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.22 DIS\_types structure CancelTable 102 DIS\_types structure Cancel

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.23 DIS\_types structure CheckFireTable 103 DIS\_types structure CheckFire

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.24 DIS\_types structure CNOTable 104 DIS\_types structure CNO

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Unused_7	Char array	7	
Leftover	Char array	16	

5.1.15.1.25 DIS\_types structure ShiftTable 105 DIS\_types structure Shift

Item Name	Type	bytes	Description
MissionStatus	Char	1	
FireForEffect	Char	1	
Unused_6	Char array	6	
Message	Char array	16	based on SHIFT MSG

5.1.15.1.26 DIS\_types structure NewMissionTable 106 DIS\_types structure NewMission

Item Name	Type	bytes	Description
MissionStatus	Char	1	
MissionType	Char	1	
Shell	Char	1	
Control	Char	1	
Fuze	Char	1	
Trajectory	Char	1	
RoundFFE	Char	1	
Unused	Char	1	
Leftover	Char array	16	

5.1.15.1.27 DIS\_types structure MTQTable 107 DIS\_types structure MTQ

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RequestAdjustment	Char	1	
EnterTarget	Char	1	
EndMission	Char	1	
Unused_4	Char array	4	
Leftover	Char array	16	

5.1.15.1.28 DIS\_types structure ShotTable 108 DIS\_types structure Shot

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RoundsFired	Char	1	
ImpactTime	Char	1	
Unused_5	Char array	5	
Leftover	Char array	16	

5.1.15.1.29 DIS\_types structure SplashTable 109 DIS\_types structure Splash

Item Name	Type	bytes	Description
MissionStatus	Char	1	
RoundsFired	Char	1	
ImpactTime	Char	1	
Unused 5	Char array	5	
Leftover	Char array	16	

5.1.15.1.30 DIS\_types structure EndOfMissionTable 110 DIS\_types structure EndOfMission

Item Name	Type	bytes	Description
MissionStatus	Char	1	
Disposition	Char	1	
Casualties	Char	1	
RecordTarget	Char	1	
CasualtyNumber	Integer	2	
Unused 2	Char array	2	
PointNumber	Char array	16	based on PT_NUM

5.1.15.1.31 DIS\_types structure Artillery Specific typeTable 111 DIS\_types structure Artillery Specific type

Item Name	Type	bytes	Description
MissionID	Char array	16	based on ID_NUM_LEN
TargetID	Char array	16	based on ID_NUM_LEN
Subtype	Union	24	one of the previous structures 5.1.15.1.21 through 5.1.15.1.30

5.1.15.1.32 DIS\_types structure Artillery typeTable 112 DIS\_types structure Artillery type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct		Artillery Specific type

5.1.15.1.33 DIS\_types structure DISArtillery typeTable 113 DIS\_types structure DISArtillery type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock type
Specific	Struct		Artillery Specific type

5.1.15.1.34 DIS\_types structure Move Specific typeTable 114 DIS\_types structure Move Specific type

Item Name	Type	bytes	Description
TargetID	Char array	16	based on ID_NUM LENG
Task	Char	1	
Who	Char	1	
When	Char	1	
Unused_3	Char array	3	
ZuluTime	Integer	2	
Where	Char array	8	based on WHERE LENG
DTG	Struct		DTG type



5.1.15.1.35 DIS\_types structure Move\_typeTable 115 DIS\_types structure Move\_type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC CommonBlock_type
Specific	Struct		Move Specific_type

5.1.15.1.36 DIS\_types structure DISMove\_typeTable 116 DIS\_types structure DISMove\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Move Specific_type

5.1.15.1.37 DIS\_types structure Status\_Specific\_typeTable 117 DIS\_types structure Status\_Specific\_type

Item Name	Type	bytes	Description
Fuel	Integer	2	
Elements	Char	1	
FailedEquip	Char array	3	based on FAILED_EQUIP
Hellfires	Char	1	
Stingers	Char	1	
Rockets	Char	1	
Rounds	Char	1	
RequestType	Char	1	
Unused_5	Char array	5	

5.1.15.1.38 DIS\_types structure Status\_typeTable 118 DIS\_types structure Status\_type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Status_Specific_type

5.1.15.1.39 DIS\_types structure DISStatus\_typeTable 119 DIS\_types structure DISStatus\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Status_Specific_type

5.1.15.1.40 DIS\_types structure Request\_Specific\_typeTable 120 DIS\_types structure Request\_Specific\_type

Item Name	Type	bytes	Description
ReportType	Char	1	
ReconType	Char	1	
Unused_6	Char array	6	

5.1.15.1.41 DIS\_types structure Request\_typeTable 121 DIS\_types structure Request\_type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct		Request_Specific_type

5.1.15.1.42 DIS\_types structure DISRequest\_typeTable 122 DIS\_types structure DISRequest\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		Request_Specific_type

5.1.15.1.43 DIS\_types structure NBC\_1Table 123 DIS\_types structure DMC\_CommonBlock\_type

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
FlashBangtime	Integer	2	
Unused_4	Char array	4	
StartDTG	Struct	16	DTG_type
EndDTG	Struct	16	DTG_type

5.1.15.1.44 DIS\_types structure NBC\_4Table 124 DIS\_types structure NBC\_4

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
DoseRate	Integer	2	
Unused_4	Char array	4	
StartDTG	Struct	16	DTG_type
EndDTG	Struct	16	DTG_type

5.1.15.1.45 DIS\_types structure NBC\_NegativeTable 125 DIS\_types structure DMC\_CommonBlock\_type

Item Name	Type	bytes	Description
Description	Char	1	
BurstType	Char	1	
DeliveredBy	Char	1	
CloudHtUnits	Char	1	
CloudWidth	Float	4	
CloudDescr	Char array	16	based on CLOUD_DESCR
CloudHeight	Integer	2	
Leftover	Char array	38	

5.1.15.1.46 DIS\_types structure NBC\_Specific\_typeTable 126 DIS\_types structure NBC\_Specific\_type

Item Name	Type	bytes	Description
Subtype	Union	64	one of the previous structures 5.1.15.1.43 through 5.1.15.1.45

5.1.15.1.47 DIS\_types structure NBC\_typeTable 127 DIS\_types structure NBC\_type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct		NBC Specific type

5.1.15.1.48 DIS\_types structure DISNBC\_typeTable 128 DIS\_types structure DISNBC\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct		NBC_Specific_type

5.1.15.1.49 DIS\_types structure MIJI Specific typeTable 129 DIS\_types structure MIJI Specific type

Item Name	Type	bytes	Description
AffectedRadioFreq	Double	8	
ProgrammedFreq	Double array	32	
KeypadEntry	Char array	16	based on KEYPAD_16
JamStartDTG	Struct	16	DTG type
JamEndDTG	Struct	16	DTG type
Description	Char	1	
PercentLost	Char	1	
Type	Char	1	
Unused_5	Char array	5	

5.1.15.1.50 DIS\_types structure MIJI typeTable 130 DIS\_types structure MIJI type

Item Name	Type	bytes	Description
Header	Struct		DMC Header type
Common	Struct		DMC CommonBlock type
Specific	Struct	96	MIJI Specific type

5.1.15.1.51 DIS\_types structure DISMIJI typeTable 131 DIS\_types structure DISMIJI type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock type
Specific	Struct	96	MIJI Specific type

5.1.15.1.52 DIS\_types structure PIREP Specific typeTable 132 DIS\_types structure PIREP Specific type

Item Name	Type	bytes	Description
Visibility	Char	1	
Restrictions	Char	1	
CloudCover	Char	1	
TurbIntensity	Char	1	
TurbFrequency	Char	1	
IcingIntensity	Char	1	
IcingType	Char	1	
WindDirection	Char	1	
WindSpeed	Integer	2	
OutsideAirTemp	Integer	2	
Barometer	Float	4	
CloudBase	Integer	4	
CloudTop	Integer	4	

5.1.15.1.53 DIS\_types structure PIREP typeTable 133 DIS\_types structure PIREP type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	24	PIREP_Specific_type

5.1.15.1.54 DIS\_types structure DISPIREP typeTable 134 DIS\_types structure DISPIREP type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	24	PIREP_Specific_type

5.1.15.1.55 DIS\_types structure DNAV\_Specific\_typeTable 135 DIS\_types structure DNAV\_Specific\_type

Item Name	Type	bytes	Description
AircraftType	Char	1	
AircraftStatus	Char	1	
PilotStatus	Char	1	
Unused_5	Char array	5	

5.1.15.1.56 DIS\_types structure DNAV\_typeTable 136 DIS\_types structure DNAV\_type

Item Name	Type	bytes	Description
Header	Struct		DMC_Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	8	DNAV_Specific_type

5.1.15.1.57 DIS\_types structure DISDNAV\_typeTable 137 DIS\_types structure DISDNAV\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	8	DNAV_Specific_type

5.1.15.1.58 DIS\_types structure FreeText\_Specific\_typeTable 138 DIS\_types structure FreeText\_Specific\_type

Item Name	Type	bytes	Description
FreeText	Char array	256	based on FREE TEXT LENG

5.1.15.1.59 DIS\_types structure FreeText\_typeTable 139 DIS\_types structure FreeText\_type

Item Name	Type	bytes	Description
Header	Struct		DMC Header_type
Common	Struct		DMC_CommonBlock_type
Specific	Struct	256	FreeText_Specific_type

5.1.15.1.60 DIS\_types structure DISFreeText\_typeTable 140 DIS\_types structure DISFreeText\_type

Item Name	Type	bytes	Description
Header_And_Common	Struct		Header_And_CommonBlock_type
Specific	Struct	256	FreeText_Specific_type

5.1.15.1.61 DIS\_types structure DMCPDU\_typeTable 141 DIS\_types structure DMCPDU\_type

Item Name	Type	bytes	Description
Variant	Union		one of the following structures :
Ack			Ack_type
Spot			Spot_type
BDA			BDA_type
Recon			Recon_type
Artillery			Artillery_type
Move			Move_type
Status			Status_type
Request			Request_type
NBC			NBC_type
MIJI			MIJI_type
PIREP			PIREP_type
DNAV			DNAV_type
FreeText			FreeText_type



5.1.15.1.62 DIS\_types structure DMCDISPDU typeTable 142 DIS\_types structure DMCDISPDU type

Item Name	Type	bytes	Description
Variant	Union		one of the following structures :
DISAck			DISAck_type
DISSpot			DISSpot_type
DISBDA			DISBDA_type
DISRecon			DISRecon_type
DISArtillery			DISArtillery_type
DISMove			DISMove_type
DISStatus			DISStatus_type
DISRequest			DISRequest_type
DISNBC			DISNBC_type
DISDISMIJI			DISMIJI_type
DISPIREP			DISPIREP_type
DISDNAV			DISDNAV_type
DISFreeText			DISFreeText_type

## 5.1.15.2 DMC\_PDU define statements

Table 143 DIS\_types define statements

Name	Value	Description
KEYPAD 4	4	
KEYPAD 8	8	
KEYPAD 16	16	
KEYPAD 32	32	
ID NUM LENG	16	
RET ID LENG	32	
WHERE LENG	8	
BITS IN BYTE	8	
SIGNAL DATA COMMON BYTES	212	
DIS HEADER SIZE	4	
DMC PDU	0xDC	
DMCDIS PDU	0x1A	
DMC OTHER PDU	0	
DMC ACKPPDU	1	
DMC SPOT PDU	2	
DMC BAD PDU	3	
DMC RECON PDU	4	
DMC ARTY PDU	5	
DMC MOVE PDU	6	
DMC STATUS PDU	7	
DMC REQUEST PDU	8	

DMC NBC PDU	9	
DMC MIJI PDU	10	
DMC PIREP PDU	11	
DMC DNAV PDU	12	
DMC FREE TEXT PDU	13	
DMC ATHS PDU	14	
NUMBER DMC PDUS	15	
DEMC HDR	8	
DMC HDR SPARE	4	
DMC COMMON	232	
DSIDMC HEADER COMMON	240	
CEOI LENG	8	
MAX CEOI RCVR	8	
ANNOTATION	64	
SIM UNUSED ARRAY SIZE	13	
UNIT LENG	16	
DMC RECON GND ROUTE PDU	0	
DMC RECON AIR ROUTE PDU	1	
DMC RECON BRIDGE PDU	2	
DMC RECON LZ PZ PDU	3	
DMC RECON BP OP PDU	4	
DMC RECON CROSSING PDU	5	
NUM DMC RECON PDUS	6	
DMC ARTY REPEAT PDU	0	
DMC ARTY CANCEL PDU	1	
DMC ARTY CHECK PDU	2	
DMC ARTY CNO PDU	3	
DMC ARTY SHIFT PDU	4	
DMC ARTY NW MSN PDU	5	
DMC ARTY MTO PDU	6	
DMC ARTY SHOT PDU	7	
DMC ARTY SPLASH PDU	8	
DMC ARTY EOM PDU	9	
DMC ARTY NEGLECT PDU	0xA	
NUM DMC ARTY PDU	0xB	
SHIFT MSG	16	
PT NUM	16	
FAILED EQUIP	3	
DMC NBC 1 PDU	0	
DMC NBC 2 PDU	1	
DMC NBC 3 PDU	2	
DMC NBC 4 PDU	3	
DMC NBC 5 PDU	4	
NUM DMC NBC PDUS	5	
CLOUD DESCR	16	
PGM FREQ	4	
FREE TEXT LENG	256	

## 5.1.16 DMC\_types.h

This header file contains definitions of various basic data elements, including how these elements are represented as communicated bits. The elements compose the fields which compose each PDU.

5.1.16.1.1 DMC\_types structure DTG\_typeTable 144 DMC\_types structure DTG\_type

Item Name	Type	bytes	Description
Hundredth	Char array	2	based on TWO_DIGITS
Second	Char array	2	based on TWO_DIGITS
Minute	Char array	2	based on TWO_DIGITS
Hour	Char array	2	based on TWO_DIGITS
Day	Char array	2	based on TWO_DIGITS
Month	Char array	2	based on TWO_DIGITS
Year	Char array	4	based on FOUR_DIGITS

5.1.16.1.2 DMC\_types structure EncodingScheme\_typeTable 145 DMC\_types structure EncodingScheme\_type

Item Name	Type	bytes	Description
Class	Integer	2	2 bit field
Scheme			14 bit field

5.1.16.1.3 DMC\_types structure UTM\_typeTable 146 DMC\_types structure UTM\_type

Item Name	Type	bytes	Description
Sector	Char array	2	based on TWO_DIGITS
Easting	Char array	7	based on UTM_MAP_COORD
Northing	Char array	7	based on UTM_MAP_COORD

5.1.16.1.4 DMC\_types structure Speed\_typeTable 147 DMC\_types structure Speed\_type

Item Name	Type	bytes	Description
Mph	Integer	2	1 bit field
Value			15 bit field

## 5.1.16.2 DMC\_types define statements

Table 148 DMC\_types define statements

Name	Value	Description
TWO_DIGITS	2	
FOUR_DIGITS	4	
ENCODESCHEME_VOICE	0	
ENCODESCHEME_RAWDATA	1	
ENCODESCHEME_APPSPEC	2	
ENCODESCHEME_POINTER	3	
UTM_MAP_COORD	7	
SPEED_IN_KPH	0	
SPEED_IN_MPH	1	
DMC_RADIOID	1	
DMC_PROTOCOLID	0xDC	
TERMINAL_UNKNOWN	0	
TERMINAL_PILOT_STN	1	
TERMINAL_CPG_STN	2	
TERMINAL_TOC_DMC	3	
TERMINAL_FSE	4	
PERSON_UNKNOWN	0	
PERSON_PILOT	1	
PERSON_CPG	2	
PERSON_G2	3	
PERSON_G3	4	
PERSON_S1	5	
PERSON_S2	6	
PERSON_FO	7	
PERSON_MO	8	
PERSON_ANY	9	
PRIORITY_ROUTINE	0	
PRIORTY_URGENT	1	
ACTIVITY_NONE	0	
ACTIVITY_STATIONARY	1	
ACTIVITY_MOVING	2	
ACTIVITY_DUG_IN	3	

ACTIVITY_RETREATING	4	
ACTIVITY_ADVANCING	5	
ACTIVITY_ATTACKING	6	
ACTIVITY_DAMAGED	7	
ACTIVITY_KILLED	8	
ACTIVITY_MOBILE_KILL	9	
TARGET_UNKNOWN	0	
TARGET_ADA	1	
TARGET_SAM	2	
TARGET_TANK	3	
TARGET_WHLD	4	
TARGET_TRKD	5	
TARGET_ACFT	6	
TARGET_TRPS	7	
TARGET_UNCL	8	
DIRECTION_N	0	
DIRECTION_NE	1	
DIRECTION_E	2	
DIRECTION_SE	3	
DIRECTION_S	4	
DIRECTION_SW	5	
DIRECTION_W	6	
DIRECTION_NW	7	
OBS_INT_OTHER	0	
OBS_INT_CONT_MSN	1	
OBS_INT_ENGA	2	
OBS_INT_HLD	3	
OBS_INT_RTS_IHLD	4	
OBS_INT_OBSV_TGT	5	
CLASSFORM_NULL	0	
CLASSFORM_W	1	
CLASSFORM_TYPE	2	
CLASSFORM_MLC	3	
CLASSFORM_OHC	4	
CLASSFORM_OB	5	
CLASSFORM_BLOCK	6	
OBSTACLES_NONE	0	
OBSTACLES_KEYPAD	1	
OBSTACLES_ADA	2	
OBSTACLES_TOWR_ANT	3	
OBSTACLES_WIRES	4	
OBSTACLES_TERR	5	
OBSTACLES_TREES	6	
OBSTACLES_BLDG	7	
OBSTACLES_NDST_CAMP	8	
OBSTACLES_OTHER	9	
BRDGTYP_UNKNOWN	0	
BRDGTYP_TRUSS	1	
BRDGTYP_GIRDER	2	

BRDGTYP_BEAM	3	
BRDGTYP_SLAB	4	
BRDGTYP_CLOS	5	
BRDGTYP_ARCHOP	6	
BRDGTYP_SUSPEN	7	
BRDGTYP_FLOATING	8	
BRDGTYP_SWING	9	
BRDGDMG_NONE	0	
BRDGDMG_LIGHT	1	
BRDGDMG_MODERATE	2	
BRDGDMG_SEVERE	3	
BRDGDMG_DESTROYED	4	
BRDGCONMAT_UNKNOWN	0	
BRDGCONMAT_KEYPAD	1	
BRDGCONMAT_METAL	2	
BRDGCONMAT_STONE	3	
BRDGCONMAT_CONCRETE	4	
ACT_LIKELY_NONE	0	
ACT_LIKELY_EXPECTED	1	
ACT_LIKELY_POSSIBLE	2	
ACT_LIKELY_UNLIKELY	3	
MSNSTAT_NULL	0	
MSNSTAT_REQUESTED	1	
MSNSTAT_READY	2	
MSNSTAT_SHOT	3	
MSNSTAT_SPLASH	4	
MSNTYPE_NULL	0	
MSNTYPE_FFE	1	
MSNTYPE_ADJ_FIRE	2	
MSNTYPE_IMMED_SUPPR	3	
MSNTYPE_SUPPR	4	
SHELL_NULL	0	
SHELL_HE	1	
SHELL_ICM	2	
SHELL_CPRHD	3	
SHELL_SMOKE	4	
SHELL_WP	5	
SHELL_CORD_ILLUM	6	
CNTL_NULL	0	
CNTL_AMC	1	
CNTL_WR	2	
CNTL_TOT	3	
FUZE_NULL	0	
FUZE_QUICK	1	
FUZE_TIME_DELAY	2	
FUZE_CONCR_PRCNG	3	
TRAJ_LOW	0	
TRAJ_HIGH	1	
TRAJ_OTHER	2	

DSPN_NULL	0	
DSPN_NOT_GIVEN	1	
DSPN_BURNING	2	
DSPN_CNO	3	
CSLT_NULL	0	
CSLT_NOT_GIVEN	1	
CSLT_GIVEN	2	
TASK_NULL	0	
TASK_TO	1	
TASK_HOLD	2	
TASK_CONT_MSN	3	
TASK_RENEZ_AT	4	
TASK_ENGA_TGT	5	
TASK_MOVING_TO	6	
TASK_HLDG_AT	7	
TASK_ARRVG_AT	8	
TASK_PASSG_THRU	9	
TASK_DEPRTG_FROM	10	
MVWHO_NULL	0	
MVWHO_MYPOS	1	
MVWHO_B2	2	
MVWHO_B3	3	
MVWHO_OP1	4	
MVWHO_OP2	5	
MVWHEN_NULL	0	
MVWHEN_IMMED	1	
MVWHEN_WHNRDY	2	
MVWHEN_AMC	3	
MVWHEN_DTG	4	
BADEQPT_NULL	0	
BADEQPT_ENG1	1	
BADEQPT_ENG2	2	
BADEQPT_GUNS	3	
BADEQPT_RADIO	4	
BADEQPT_DSGNTR	5	
BADEQPT_RKTPOD	6	
BADEQPT_MSLSTN	7	
BADEQPT_RADAR	8	
BADEQPT_LASER	9	
REPTTYP_NONE	0	
REPTTYP_STATUS	1	
REPTTYP_SPOIT	2	
REPTTYP_RECON	3	
REPTTYP_MOVEMENT	4	
RECTYP_NONE	0	
RECTYP_GND_RTE	1	
RECTYP_AIR_RTE	2	
RECTYP_LZ_PZ	3	
RECTYP_BP_OP	4	

RECTYP_CROSSING	5	
BURST_NONE	0	
BURST_UNKNOWN	1	
BURST_SURFACE	2	
BURST_AIR	3	
BURST_GND	4	
DLVRY_UNKNOWN	0	
DLVRY_ARTY	1	
DLVRY_MORTAR	2	
DLVRY_ROCKET	3	
DLVRY_MISSLE	4	
DLVRY_BOMB	5	
CLDHT_UNUSED	0	
CLDHT_DEGREES	1	
CLDHT_HT_IN_FT	2	
CLDHT_HT_IN_M	3	
NBCDESC_UNUSUAL	0	
NBCDESC_INITIAL	1	
NBCDESC_INCREASING	2	
NBCDESC DECREASING	3	
NBCDESC_PeAD	4	
NBCDESC_SPECIAL	5	
NBCDESC_SERIES	6	
NBCDESC_VERIFICATION	7	
NBCDESC_SUMMARY	8	
MIJIDESC_UNKNOWN	0	
MIJIDESC_INTERMIT	1	
MIJIDESC_CONT	2	
MIJIDESC_IMITATIVE	3	
MIJIDESC_WARBLING	4	
MIJIDESC_MUSIC	5	
MIJIDESC_KEYPAD	6	
MIJITYPE_UNKNOWN	0	
MIJITYPE_MEACONING	1	
MIJITYPE_INTRUSION	2	
MIJITYPE_JAMMING	3	
MIJITYPE_INTERFERE	4	
VISI_UNLIMITED	0	
VISI_HALF_MILE	1	
VISI_ONE_MILE	2	
VISI_TWO_MILE	3	
VISI_THREE_MILE	4	
VISI_FOUR_MILE	5	
VISI_FIVE_MILE	6	
VISI_RESTR_NONE	0	
VISI_RESTR_HAZE	1	
VISI_RESTR_RAIN	2	
VISI_RESTR_SNOW	3	
VISI_RESTR_FOG	4	



VISI_RESTR_SMOKE	5	
VISI_RESTR_DRIZZLE	6	
VISI_RESTR_DUST	7	
VISI_RESTR_BLIZZARD	8	
CLDCVR_NONE	0	
CLDCVR_OVERCAST	1	
CLDCVR_BROKEN	2	
CLDCVR_SCATTERED	3	
TRBINT_NONE	0	
TRBINT_LIGHT	1	
TRBINT_MODERATE	2	
TRBINT_SEVERE	3	
TRBINT_EXTREME	4	
TRBFREQ_NONE	0	
TRBFREQ_OCCASION	1	
TRBFREQ_INTERMIT	2	
TRBFREQ_CONT	3	
ICEINT_NONE	0	
ICEINT_TRACE	1	
ICEINT_LIGHT	2	
ICEINT_MODERATE	3	
ICEINT_SEVERE	4	
ICETYP_NULL	0	
ICETYP_RIME	1	
ICETYP_CLEAR	2	
ACRFT_TYP_UNKNOWN	0	
ACRFT_TYP_RAH66	1	
ACRFT_TYP_AH64	2	
ACRFT_TYP_OH58	3	
ACRFT_TYP_AH1	4	
ACRFT_TYP_UH1	5	
ACRFT_TYP_CH47	6	
ACRFT_TYP_OV1	7	
ACRFT_TYP_OH6	8	
ACRFT_TYP_CH54	9	
ACRFT_TYP_UH60	10	
ACRFT_TYP_RPV	11	
ACRFT_STAT_UNKNOWN	0	
ACRFT_STAT_RECOVER	1	
ACRFT_STAT_DAMAGED	2	
ACRFT_STAT_DESTROYED	3	
PILOT_STAT_UNKNOWN	0	
PILOT_STAT_GOOD	1	
PILOT_STAT_WIA	2	
PILOT_STAT_KIA	3	
PILOT_STAT_POW	4	

## 5.1.17 dms.h

This header file contains parameters for the Digital Message Server routines.

5.1.17.1.1 dms structure msgdests\_tTable 149 dms structure msgdests\_t

Item Name	Type	bytes	Description
name	Char array	8	based on MAXDESTSIZE

5.1.17.1.2 dms structure dms\_shmem\_tTable 150 dms structure dms\_shmem\_t

Item Name	Type	bytes	Description
nbr_readers	Integer	4	
dests	Array of Struct	64	msgdest_t
exercise_id	Char	1	
pdu_length	Integer	4	
pdu	Char array	1500	based on MAXPDUSIZE

5.1.17.1.3 dms structure dms\_outgoingmsg\_tTable 151 dms structure dms\_outgoingmsg\_t

Item Name	Type	bytes	Description
msg_type	Integer	4	
dm_size	Integer	4	
dm	Char array	1500	based on MAXPDUSIZE

5.1.17.1.4 dms structure logTable 152 dms structure log

Item Name	Type	bytes	Description
pid	Integer	4	
exercise_id	Char	1	
dests	Array of Struct	64	msgdests_t

5.1.17.1.5 dms structure dms\_msg\_tTable 153 dms structure dms\_msg\_t

Item Name	Type	bytes	Description
msg_type	Integer	4	
msg	Union	69	either shmem_addr or log Struct - 5.1.17.1.4

## 5.1.17.2 dms define statements

Table 154 dms define statements

Name	Value	Description
MAXMSGDESTS	8	# of names in PDU req
MAXDESTSIZE	8	size of name/group
MAX CLIENTS	8	
MAXPDU	32	
MAXPDUSIZE	1500	
MSG_LOGIN	1	
MSG_LOGOUT	2	
MSG_DMAVAIL	3	
MSG_DONE	4	
MSG_NEWDESTS	5	
MSG_FREE	6	
MSG_CLEANUP	7	
DMC_MAXMSGTYPE	10	
DMS_ENET_RECEIVE	1	
DMS_ENET_TRANSMIT	2	
FAN_OUT_Q_KEY		((key_t) 54321
OUTGOING_Q_KEY		((key_t) 54322
INCOMING_SHMEM_KEY		((key_t) 54323
DMS_PERMS	0666	

### 5.1.18 fcn\_proto.h

In C, function prototyping (the defining the input arguments and resultant output from the function) is required. The compiler will check the prototyping of a function against the actual arguments used within the source code of the calling function and flag as a compiler error any incongruities. This header file prototypes all of the general use functions used in the DMC.

### 5.1.19 global\_netif.h

This header file defines the globally used parameters for use by the ethernet interface functions, identifies and dimensions external data entities and prototypes the DIS and SIM network interface functions. Table 5.1.19.1 structures is not applicable.

## 5.1.19.2 global\_netif define statements

Table 155 global\_netif define statements

Name	Value	Description
DSAP	0xaa	
SSAP	0xaa	
CNTL	0x03	
SIM_SOCKET_NAME1	soc_1	
SIM_PROTOCOL_ID	0x08000800	
SIM_PROTOCOL_ID_B1	0x08	
SIM_PROTOCOL_ID_B2	0x00	
SIM_PROTOCOL_ID_B3	0x08	
SIM_ETHER_TYPE_B1	0x52	
SIM_ETHER_TYPE_B2	0x08	
SIM_BROADCAST_TYPE	0xFF	
DIS_BROADCAST_ADDR	0x89F9200	
DIS_PROTOCOL_ID	0x00000800	
DIS_PROTOCOL_ID_B1	0x00	
DIS_PROTOCOL_ID_B2	0x00	
DIS_PROTOCOL_ID_B3	0x00	
DIS_ETHER_TYPE_B1	0x52	
DIS_ETHER_TYPE_B2	0x09	
HEADER	0	
DATA	1	
MAXIOV	2	
TEST_PDU_FN_SIZE	40	
SITE_SIMULATOR_BYTES	4	
ETHER_TYPE_POS	6	
PROTOCOL_ID_POS	3	
READONLY	0	
SYS_ETHER_ADDR	"ETHER"	
SYS_IP_ADDR	"INET"	
SYS_IP_BROADCAST_ADDR	"BROADCAST"	

## 5.1.20 global\_vars.h

This header file defines the globally used parameters for use by the DMC functions, identifies and dimensions external data entities and prototypes the PDU decode functions. Table 5.1.20.1 structures is not applicable.

## 5.1.20.2 global\_vars define statements

Table 156 global\_vars define statements

Name	Value	Description
TRUE	1	
FALSE	0	
UNUSED	0	
SIGREC	SIGUSR1	
SIGXMT	SIGUSR2	
USEC_PER_MS	1000	
MSEC_PER_SEC	1000	
MSEC_PER_HOUR	3600000	
USEC_PER_SEC	1000000.0	USEC_PER_MS*1000
SEC_PER_HOUR	3600	
USEC_PER_HOUR	3600000000	USEC_PER_SEC*SEC_PER_HOUR
DIS_TIME_UNITS	(double)0x80000000	= 0.5965
BAMS_PER_DEGREE	(float)0x80000000/ (float)180	

## 5.1.21 netif.h

This header file defines the globally used parameters for use by the network interface functions in sizing arrays and pointing to particular bytes within a PDU.

5.1.21.1.1 netif structure Int ArrayTable 157 dms structure Int Array

Item Name	Type	bytes	Description
ethernet_address	Integer array	24	based on ENETADDRSZ

## 5.1.21.2 netif define statements

Table 158 netif define statements

Name	Value	Description
MAX_IFNAME_SIZE	10	
ETHER_HDR_SIZE	14	
DISADDRSZ	4	
ENETADDRSZ	6	
MAX_LINE_WIDTH	132	
ETHERTYPE_SIM	0x5208	
ETHERTYPE_DIS	0x5209	
PROMISCUOUS_MODE	0x01	

## 5.2 DMCC Include Files

The following table maps the DMCC CSCI include files for interprocess communication, the PDU Builder, and the digital message server, and CSC's and CSU's that use them:

Table 159 DMCC CSC/CSU Data File Map

CSC	CSU's	INCLUDE FILENAME	COMMENTS
dms	All	dms.h	Contains parameters and data structures for inter process communication
build pdu	All	dms.h	Contains parameters and data structures for inter process communication
ipc	All	dms.h	Contains parameters and data structures for inter process communication

## **6. CSCI data files.**

**6.1. The DMCC CSCI uses one external data file which contains an integer. This integer expresses the difference in minutes between Local Time and Zulu Time (also known as Greenwich Mean Time, or GMT).**

**Name:**

**`"/tmp/dmcc_timezone_offset"`**

**Format:**

**Contains a single integer value**

**Location:**

**Maintained in temporary files directory directory: `/tmp`.**

**Description:**

**The `dmcc_timezone_offset` file is a block oriented file containing a single integer. The contents of the file are ascii readable. The value represents the number of minutes that local time is ahead of Zulu time (that is, the number of minutes to be subtracted from local time to obtain Zulu). If the value of this integer is positive, then local time is ahead of Zulu time; if the value is negative, then local time is "behind" Zulu time.**

**GMT = ZULU.**



## **7. Requirements traceability.**

Requirements traceability are covered under Section 4, Preliminary Design.

## **8. Notes.**

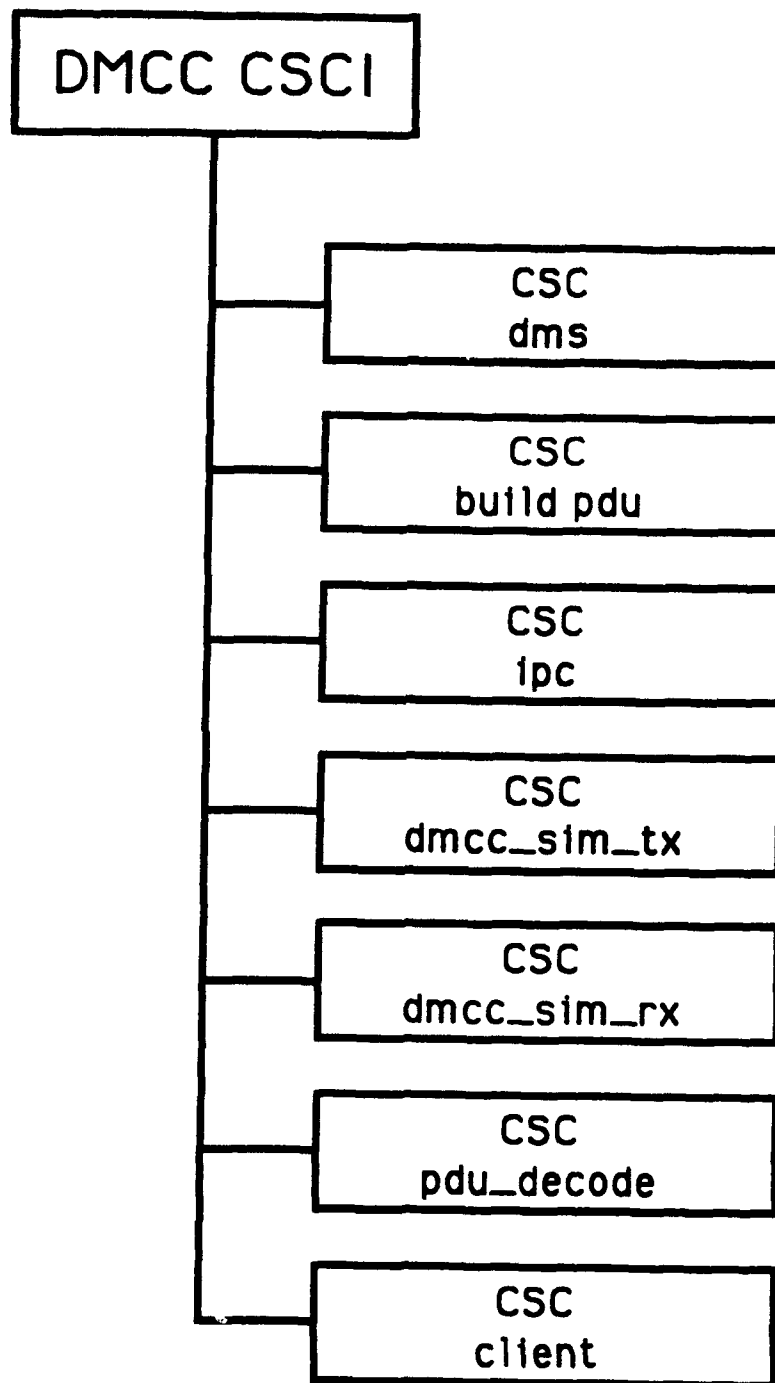
### **GLOSSARY and ACRONYMS**

<b>DMCC:</b>	<b>Digital Message Communications Console</b>
<b>DMC:</b>	<b>Digital Message Communications</b>
<b>DMCS:</b>	<b>Digital Message Communications System</b>
<b>CEOI:</b>	<b>Communications Electronic Operational Identifier (Call Sign)</b>
<b>Sys Main:</b>	<b>System Main</b>
<b>SMD:</b>	<b>Situation Management Display</b>
<b>TMI:</b>	<b>Tactile Message Indicator</b>
<b>CIK:</b>	<b>Cockpit Interactive Keyboard</b>
<b>MOVCMD:</b>	<b>Movement Command</b>
<b>IPC:</b>	<b>Unix System V Interprocess Communications</b>
<b>GUI:</b>	<b>Graphical User Interface</b>
<b>MTO:</b>	<b>Message To Observer</b>
<b>CNO:</b>	<b>Can Not Observe</b>
<b>ARTY:</b>	<b>Artillery</b>
<b>BDA:</b>	<b>Battle Damage Assessment</b>
<b>PIREP:</b>	<b>Pilot Report</b>
<b>ENET:</b>	<b>Ethernet</b>
<b>SIMNET:</b>	<b>Simulation Network</b>
<b>DIS:</b>	<b>Distributed Interactive Simulation</b>
<b>PDU:</b>	<b>Protocol Data Unit</b>
<b>APDU:</b>	<b>Association Layer Protocol Data Unit</b>
<b>X</b>	<b>X-Windows Windowing System</b>
<b>OSF</b>	<b>Open Software Foundation</b>
<b>BX</b>	<b>Builder's Xcessory</b>
<b>CSC:</b>	<b>Computer Software Component</b>
<b>CSU:</b>	<b>Computer Software Unit</b>
<b>DMS:</b>	<b>Digital Message Server</b>
<b>NBC:</b>	<b>Nuclear, Biological, &amp; Chemical Weapons</b>
<b>DTG:</b>	<b>Date Time Group</b>
<b>UTM:</b>	<b>Universal Transverse Mercator</b>
<b>CSCI:</b>	<b>Computer Software Configuration Item</b>

## **10. Appendix A - DMCC Software Structure Charts.**

The structure charts on this and the following pages depict collectively the overall structure of the DMCC CSCI.

## 10.1 DMCC CSCI

**Figure 13** Top Level DMCC CSCI Structure

## 10.2 CSC dms

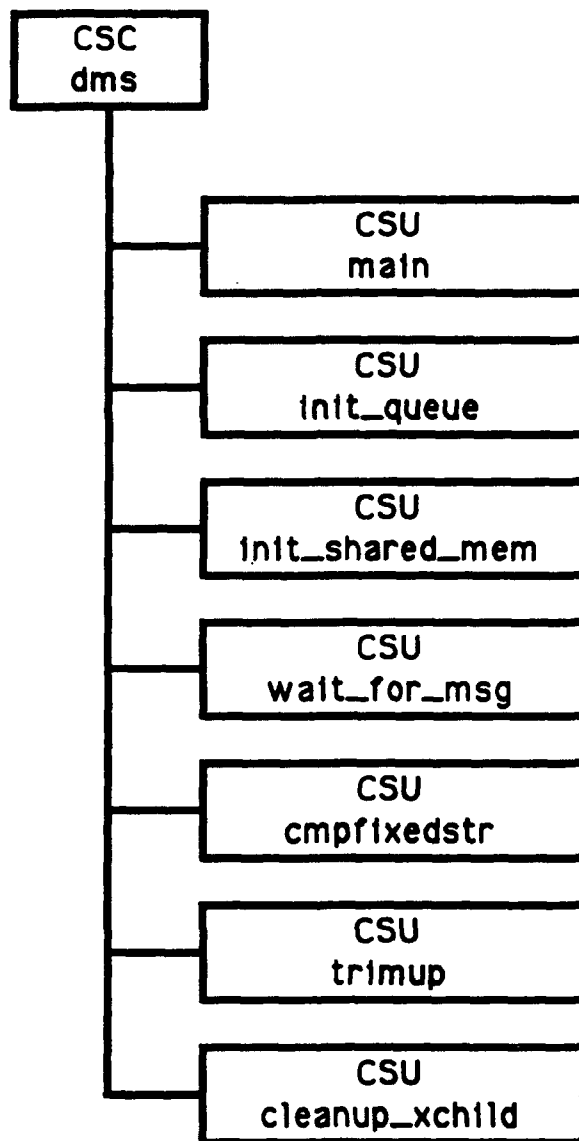


Figure14 CSC dms Structure

## 10.3 CSC build\_pdu

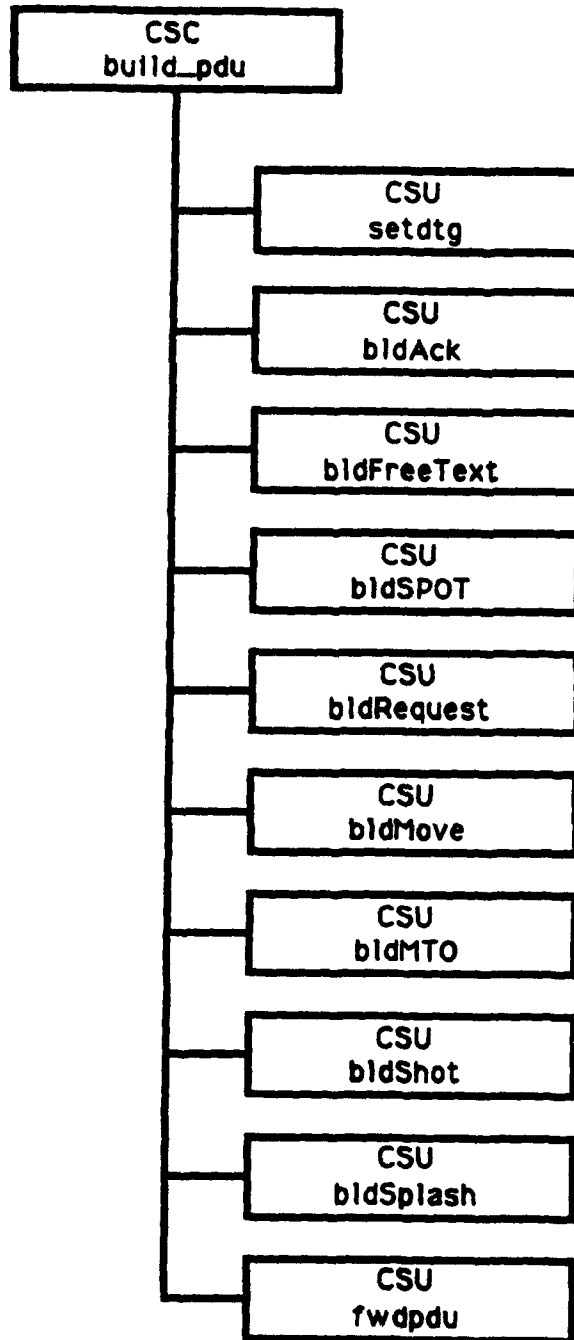


Figure 15 CSC build\_pdu Structure

## 10.4 CSC ipc

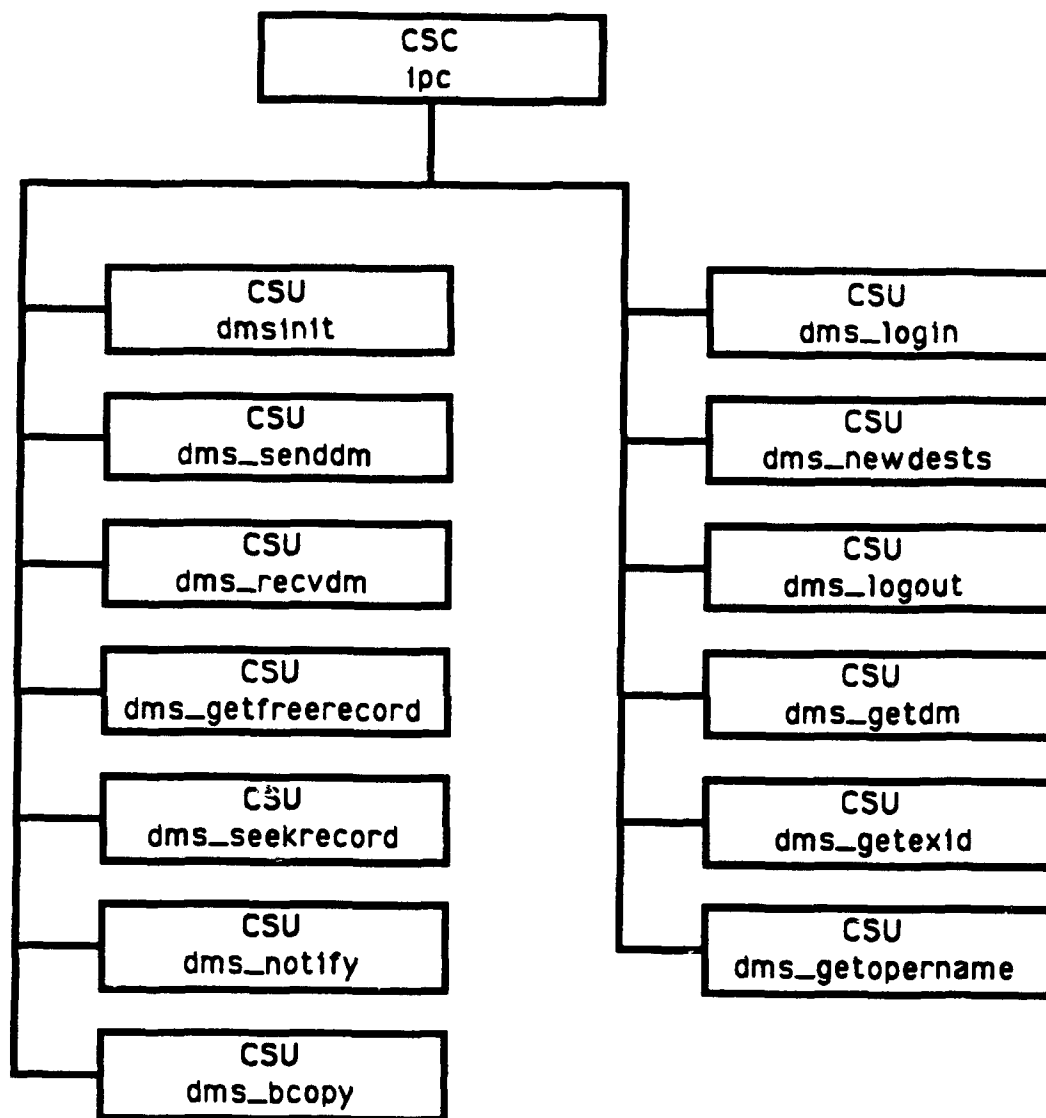


Figure 16 CSC ipc Structure

## 10.5 CSC dmcc\_sim\_tx

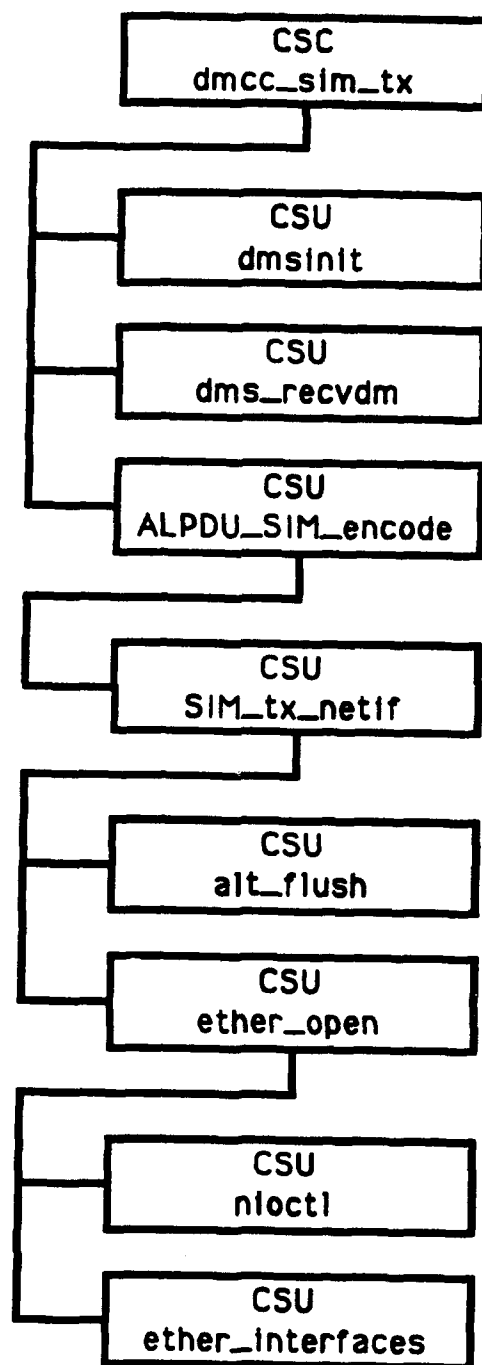


Figure 17 Sublevel CSC dmcc\_sim\_tx Structure

## 10.6 Sublevel CSC dmcc\_sim\_rx

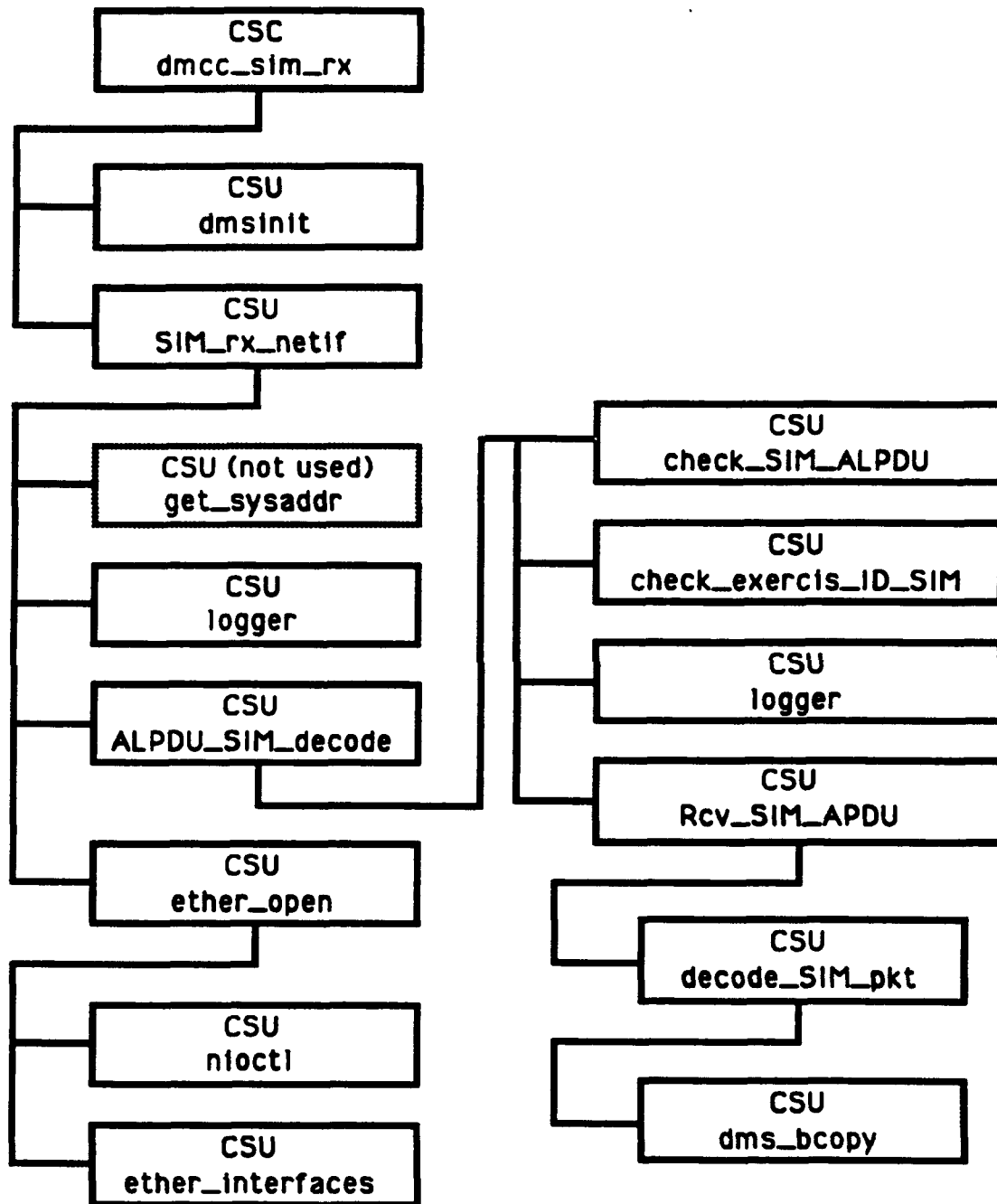


Figure 18 Sublevel CSC dmcc\_sim\_rx Structure



## 10.7 CSC pdu\_decode

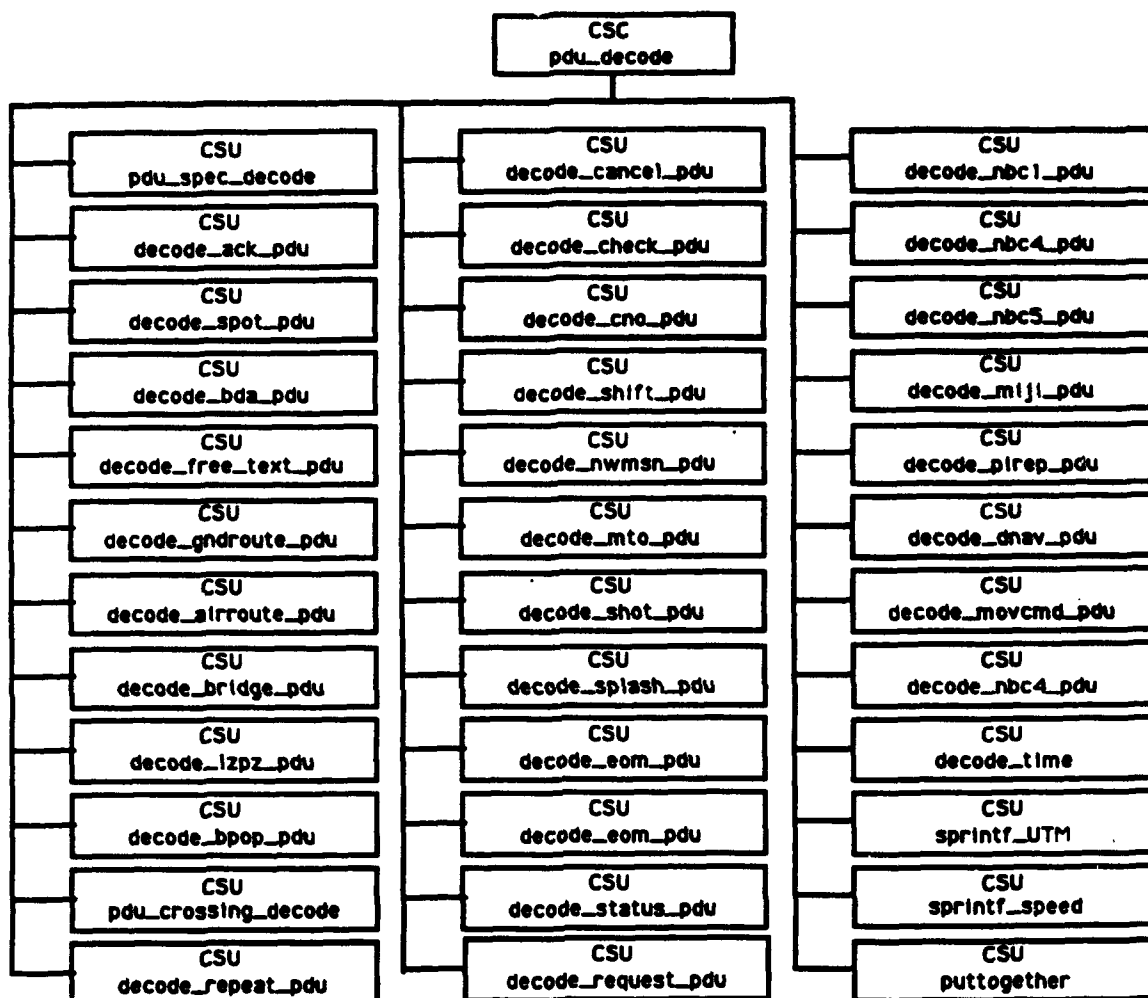


Figure 19 CSC pdu\_decode Structure

## 10.8 CSC client

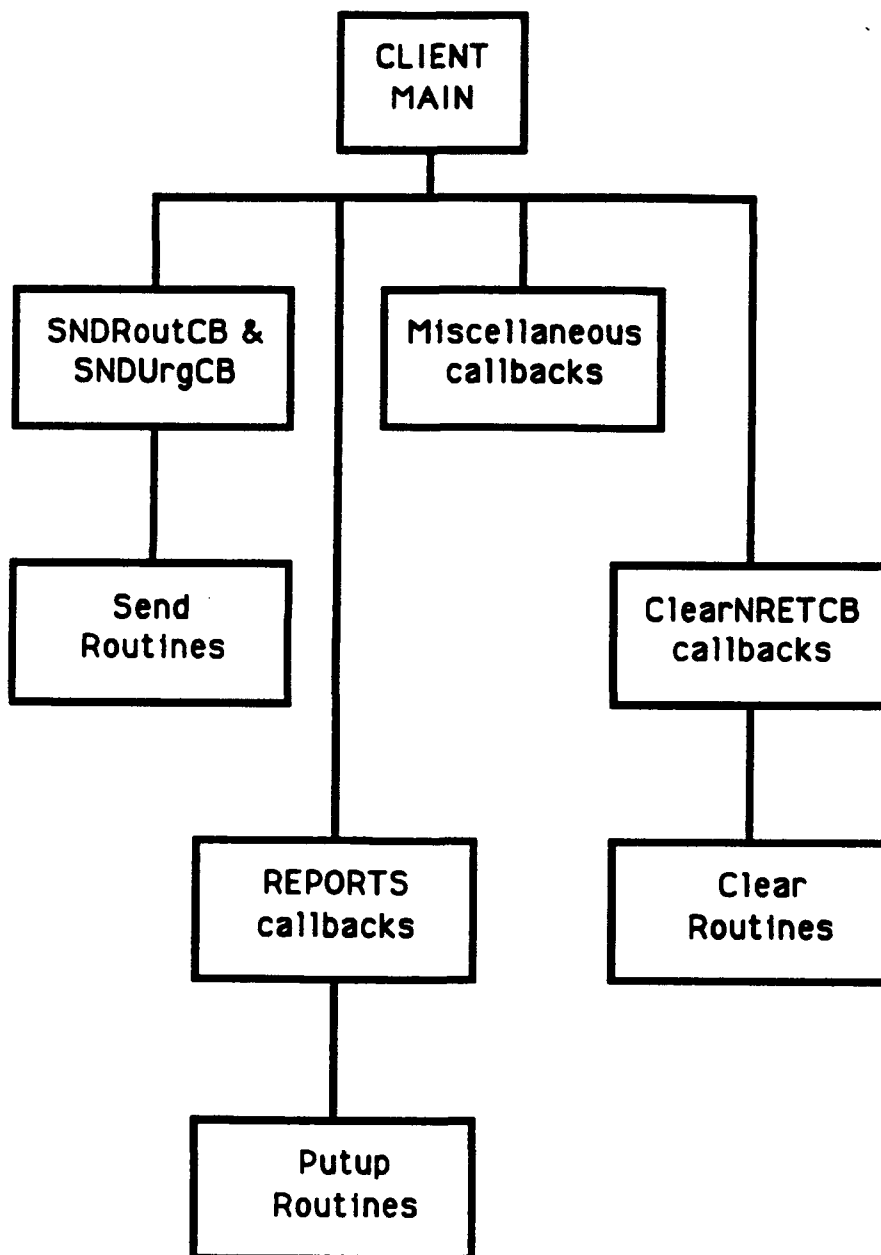


Figure 20 CSC client Structure

## **20. Appendix B - DMCC "C" Source Data Structures.**

This is the set of "C" language header files (\*.h) which describe the various datastructures and other data files . These files are located in the ADST Configuration Management Directory, under the following pathnames

a3/adst-cm/dmcc/include

a3/adst-cm/dmcc/gui/include

### **30. Appendix C - DMCC Client State Transition Diagram.**

The diagram on the following pages shows the various states the CLIENT process can occupy, and the various actions which cause the process to change states. This diagram is an Excel 3.0 spreadsheet found in the MCC Software Development File under pathname IRM/ADST MacIIci #2/MCC SDF/Deliverable Documents/DMCC State Transition Diagram.

The X-Client Process State Event Matrix is shown on the following pages. This diagram shows actions and transitions from various window states bound to event flags representing operator intervention, such as clicking on a button region of a window. Each cell shows what actions are taken as a result of an event followed by a semicolon (;) character, which is followed by an indication of the next window to transition to (make active), if any. The top row is a list of the window events and the left column is a list of window states.

Table 160 Client State Transition Table

	Logon to Network Selected	Stand Alone Selected	Sys Main Selected	Address List Selected
Logon Window Active	Logon name and exercise ID to Network; go to SYS MAIN	No Logon; Go to SYS MAIN	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	Go To Address List Window
Address List Edit Window Active	n/a	n/a	Record Address List; Go to Sys Main	n/a
Location List Edit Window Active	n/a	n/a	Record Location List; Go to Sys Main	n/a
CEOI / Group List Edit Window Active	n/a	n/a	Record CEOI / Group List; Go to Sys Main	n/a
Sys Main Msg Window Active	n/a	n/a	Go to Sys Main	n/a
Sys Main Msg Read Window Active	n/a	n/a	Go to Sys Main	n/a
Sys Main Rprt Window Active	n/a	n/a	Go to Sys Main	n/a

Table 160 Client State Transition Table, cont.

	CEOI/ Group List Selected	Location List Selected	MSG Selected	RPRT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	Go to CEOI/ Group List Window	Go To Location List Window	Go to Sys Main Msg Window	Go To RPRT Window
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Up Selected	Down Selected	Add Selected	Delete Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	Round-Robin address highlight	Round-Robin address highlight	Add Address to Address List	Delete selected Address from Address List
Location List Edit Window Active	Round-Robin Location highlight	Round-Robin Location highlight	Add Location to Location List	Delete selected Location from Location List
CEOI / Group List Edit Window Active	Round-Robin Group highlight	Round Robin Group highlight	Add Group Name to Group List	Delete Group Name from Group List
Sys Main Msg Window Active	n/a	n/a	n/a	Delete Selected Message
Sys Main Msg Read Window Active	n/a	n/a	n/a	Delete Shown Message
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	READ Selected	Logout Selected	Prev Selected	Next Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	Logout, shut down client process	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	Retrieve message; go to Sys Main Msg Read	n/a	Round-Robin message highlight	Round-Robin message highlight
Sys Main Msg Read Window Active	Go to Sys Main Msg Window	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a



Table 160 Client State Transition Table, cont.

	Re-Use Selected	Reply Selected	Save & Exit Selected	Save & Return Selected
Lagon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	Mark currently selected message for re-use; go to Sys Main Rpts	Mark currently selected message for reply; go to Sys Main Rpts	n/a	n/a
Sys Main Msg Read Window Active	Mark displayed message for re-use; go to Sys Main Rpts	Mark currently selected message for reply; go to Sys Main Rpts	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	Do not delete entered partial message; Go to SYS MAIN	n/a

Table 160 Client State Transition Table, cont.

	Clear & Return Selected	SPOT Selected	MTO Selected	SHOT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	Go to Spot Window	Go to MTO Window	Go to SHOT window

Table 160 Client State Transition Table, cont.

	<b>SPLASH Selected</b>	<b>FREE TEXT Selected</b>	<b>REQT Selected</b>	<b>MOVCMD Selected</b>
<b>Logon Window Active</b>	n/a	n/a	n/a	n/a
<b>Sys Main Window Active</b>	n/a	n/a	n/a	n/a
<b>Address List Edit Window Active</b>	n/a	n/a	n/a	n/a
<b>Location List Edit Window Active</b>	n/a	n/a	n/a	n/a
<b>CEOI / Group List Edit Window Active</b>	n/a	n/a	n/a	n/a
<b>Sys Main Msg Window Active</b>	n/a	n/a	n/a	n/a
<b>Sys Main Msg Read Window Active</b>	n/a	n/a	n/a	n/a
<b>Sys Main Rprt Window Active</b>	Go to SPLASH window	Go to FREE TEXT window	Go to REQUEST window	Go to MOVCMD window

Table 160 Client State Transition Table, cont.

	SND ROUT Selected	SND URG Selected	ADRS Selected	ADRS MORE Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEO: / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	ENEMY TYPE Selected	ENEMY ACTIVITY Selected	ENEMY ACTIVITY MORE Selected	DIREC Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / C... List Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	OBS INT selected	SPEED Entered	NUMBER Selected	UNIT Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	REQ ADJ Selected	EAT Selected	EOM Selected	TASK Selected
Logon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	TASK MORE Selected	WHEN Selected	LCTN Selected	LCTN MORE Selected
Lagon Window Active	n/a	n/a	n/a	n/a
Sys Main Window Active	n/a	n/a	n/a	n/a
Address List Edit Window Active	n/a	n/a	n/a	n/a
Location List Edit Window Active	n/a	n/a	n/a	n/a
CEOI / Group List Edit Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Window Active	n/a	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a	n/a



Table 160 Client State Transition Table, cont.

	UTM Entered	TIME Entered	FREE TEXT ANNOTATIO N Entered
Logon Window Active	n/a		n/a
Sys Main Window Active	n/a		n/a
Address List Edit Window Active	n/a		n/a
Location List Edit Window Active	n/a		n/a
CEOI / Group List Edit Window Active	n/a		n/a
Sys Main Msg Window Active	n/a	n/a	n/a
Sys Main Msg Read Window Active	n/a	n/a	n/a
Sys Main Rprt Window Active	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Logon to Network Selected	Stand Alone Selected	Sys Main Selected	Address List Selected
MTD Window Active	n/a	n/a	Go to Sys Main	n/a
SHOT Window Active	n/a	n/a	Go to Sys Main	n/a
SPLASH Window Active	n/a	n/a	Go to Sys Main	n/a
MOVCMO Window Active	n/a	n/a	Go to Sys Main	n/a
REQUEST Window Active	n/a	n/a	Go to Sys Main	n/a
SPOT Window Active	n/a	n/a	Go to Sys Main	n/a
FREE TEXT Window Active	n/a	n/a	Go to Sys Main	n/a

Table 160 Client State Transition Table, cont.

	CEOI/ Group List Selected	Location List Selected	MSGs Selected	RPRT Selected
MTOWindow Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMDD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Up Selected	Down Selected	Add Selected	Delete Selected
MTD Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMO Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	READ Selected	Logout Selected	Prev Selected	Next Selected
MTD Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	Re-Use Selected	Reply Selected	Save & Exit Selected	Save & Return Selected
MTO Window Active	n/a	n/a	n/a	Do not delete entered partial MTO message; Go to SYS MAIN RPRTS
SHOT Window Active	n/a	n/a	n/a	Do not delete entered partial SHOT message; Go to SYS MAIN RPRTS
SPLASH Window Active	n/a	n/a	n/a	Do not delete entered partial SPLASH message; Go to SYS MAIN RPRTS
MOVCMD Window Active	n/a	n/a	n/a	Do not delete entered partial MOVCMD message; Go to SYS MAIN RPRTS
REQUEST Window Active	n/a	n/a	n/a	Do not delete entered partial REQUEST message; go to SYS MAIN RPRTS
SPOT Window Active	n/a	n/a	n/a	Do not delete entered partial SPOT message; Go to SYS MAIN RPRTS
FREE TEXT Window Active	n/a	n/a	n/a	Do not delete entered partial FREE TEXT message; Go to SYS MAIN RPRTS

Table 160 Client State Transition Table, cont.

	Clear & Return Selected	SPOT Selected	MTO Selected	SHOT Selected
MTO Window Active	Delete entered partial message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
SHOT Window Active	Delete entered partial SHOT message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
SPLASH Window Active	Delete entered partial SPLASH message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
MOVCMD Window Active	Delete entered partial MOVCMD message; Go to SYS MAIN RPRTS	n/a	n/a	n/a
REQUEST Window Active	Delete entered partial REQUEST message; go to SYS MAIN RPRTS	n/a	n/a	n/a
SPOT Window Active	Delete entered partial SPOT message; go to SYS MAIN RPRTS	n/a	n/a	n/a
FREE TEXT Window Active	Delete entered partial FREE TEXT message; Go to SYS MAIN RPRTS	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	<b>SPLASH Selected</b>	<b>FREE TEXT Selected</b>	<b>REQT Selected</b>	<b>MOVCMD Selected</b>
<b>MTO Window Active</b>	n/a	n/a	n/a	n/a
<b>SHOT Window Active</b>	n/a	n/a	n/a	n/a
<b>SPLASH Window Active</b>	n/a	n/a	n/a	n/a
<b>MOVCMD Window Active</b>	n/a	n/a	n/a	n/a
<b>REQUEST Window Active</b>	n/a	n/a	n/a	n/a
<b>SPOT Window Active</b>	n/a	n/a	n/a	n/a
<b>FREE TEXT Window Active</b>	n/a	n/a	n/a	n/a



Table 160 Client State Transition Table, cont.

	<b>SND ROUT Selected</b>	<b>SND URG Selected</b>	<b>ADRS Selected</b>	<b>ADRS MORE Selected</b>
<b>MTO Window Active</b>	Send MTO Report ROUTINE; Go to SYS MAIN RPRTS	Send MTO Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>SHOT Window Active</b>	Send SHOT Report ROUTINE; Go to SYS MAIN RPRTS	Send SHOT Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>SPLASH Window Active</b>	Send SPLASH Report ROUTINE; Go to SYS MAIN RPRTS	Send SPLASH Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>MOVCMO Window Active</b>	Send MOVCMO Report ROUTINE; Go to SYS MAIN RPRTS	Send MOVCMO Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>REQUEST Window Active</b>	Send REQUEST Report ROUTINE; Go to SYS MAIN RPRTS	Send REQUEST Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>SPOT Window Active</b>	Send SPOT Report ROUTINE; Go to SYS MAIN RPRTS	Send SPOT Report URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections
<b>FREE TEXT Window Active</b>	Send FREE TEXT REPORT ROUTINE; Go to SYS MAIN RPRTS	Send FREE TEXT REPORT URGENT; Go to SYS MAIN RPRTS	Round-Robin the ADRS highlight	Toggle alternate address selections

Table 160 Client State Transition Table, cont.

	ENEMY TYPE Selected	ENEMY ACTIVITY Selected	ENEMY ACTIVITY MORE Selected	DIREC Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	Round-Robin the ENEMY TYPE highlight	Round-Robin ENEMY ACTIVITY highlight	Toggle ENEMY ACTIVITY selections	Round-Robin the DIRECTION highlight
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	OBS INT selected	SPEED Entered	NUMBER Selected	UNIT Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMO Window Active	n/a	n/a	n/a	n/a
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	Round-Robin OBSERVER INTENT highlight	Capture SPEED Text	Capture NUMBER Text	Capture UNIT Text
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	REQ ADJ Selected	EAT Selected	EOM Selected	TASK Selected
MTOWindow Active	Set MTO message type to REQ ADJ	Set MTO message type to Enter As Target	Set MTO message type to End of Mission	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMDD Window Active	n/a	n/a	n/a	Round-Robin TASK highlight
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	TASK MORE Selected	WHEN Selected	LCTN Selected	LCTN MORE Selected
MTO Window Active	n/a	n/a	n/a	n/a
SHOT Window Active	n/a	n/a	n/a	n/a
SPLASH Window Active	n/a	n/a	n/a	n/a
MOVCMD Window Active	Toggle TASK selections	Round-Robin TASK selections	Round-Robin LCTN highlight	Toggle LCTN selections
REQUEST Window Active	n/a	n/a	n/a	n/a
SPOT Window Active	n/a	n/a	n/a	n/a
FREE TEXT Window Active	n/a	n/a	n/a	n/a

Table 160 Client State Transition Table, cont.

	UTM Entered	TIME Entered	FREE TEXT ANNOTATION Entered
MTO Window Active	n/a	n/a	Capture FREE TEXT Annotation
SHOT Window Active	n/a	n/a	Capture FREE TEXT Annotation
SPLASH Window Active	n/a	n/a	Capture FREE TEXT Annotation
MOVCMD Window Active	Capture UTMs	Capture TIME; change to ZULU if entered in LOCAL	Capture FREE TEXT Annotation
REQUEST Window Active	n/a	n/a	Capture FREE TEXT Annotation
SPOT Window Active	n/a	n/a	Capture FREE TEXT Annotation
FREE TEXT Window Active	n/a	n/a	Capture FREE TEXT Annotation

#### **40. Appendix D - DMCC Protocol Data Units (PDUs).**

The diagrams on the following pages graphically detail the message protocol data units used by the Digital Message Communications System. The DMCC software is capable of building, transmitting, receiving, decoding and displaying the SPOT, MTO, SHOT, SPLASH, REQUEST, MOVCMO, and FREE TEXT message PDUs. In addition, it is capable of receiving, decoding and displaying the BDA, RECON, ARTILLERY REPEAT, ARTILLERY CANCEL, ARTILLERY CNO ARTILLERY SHIFT, ARTILLERY NEW MISSION, ARTILLERY END MISSION, STATUS, NBC, PIREP, DNAV and MIJI message PDUs.

## 40.1 Simnet Header and common block

D C DMC PDU: Header &  
Common Block for SimnetRev. Date:  
1/6/93

32767  
 9 = Bytes (default if not specified)  
 3 = bits  
 all values specified in hex  
 fields are unsigned if not specified

## Header.HeaderType.SIM:

Header (8)	Simulation Protocol Version (1)	1 Aug 86
	PDU Kind (1)	2 Jan 90
	Exercise ID (1)	3 Jan 90 Connected
	Unused (5)	

## Common (232):

SenderID (8)	Site (2)	
	Host (2)	
	Entity (2)	
RadiolD (2)		
ExerciseTime (4)	sec since 0000 Jan . 970 GMT	
Unused (13)		
TimesForwarded (1)		
SenderTerminal (1)		
SenderPerson (1)		
Altitude (4)	ft (signed integer)	
TargetID (8)	Site (2)	
	Host (2)	
	Entity (2)	
TargetTerminal (1)		
TargetPerson (1)		
SenderCEIC (8)	16 character string	
LastFwdCEIC (8)	16 character string	
TargetCEIC (84)	168 character string	
TransmitDTS (5)	(See DTS Type)	
Transmit Frequency (8)	Hz (540 float)	
Transmit Location (5)	(See LTM Type)	
FreeText (54)	154 character string	
ZuluTime (2)	min ahead of Zulu	
SerialNumber (2)		
Priority (1)		
Overlay (1)		
Subtype (1)		
Variation (1)		

0 Unknown  
 1 Pilot Stn  
 2 CP3 Stn  
 3 TCC DMC  
 4 FSE

0 Unknown  
 1 Pilot  
 2 CP3  
 3 C2  
 4 C3  
 5 S1  
 6 S2  
 7 FC  
 8 MC  
 9 Any

Communications Electronic Operations Instructions (Call sign)

Date Time Group (DTS) Type (ASCII numbers)

Unidirection, 21	Second, 21	Minute, 21	Hour, 21	Day, 21	Month, 21	Year, 41
------------------	------------	------------	----------	---------	-----------	----------

Universal Transverse Mercator (UTM) Type (ASCII character)

Sector, 21	Easting, 71	Northing, 71
------------	-------------	--------------

0 Routine  
 1 Urgent

0 No Overlay

0 Standard  
 (unless otherwise specified in Specific)

1 ACK  
 2 SPCT  
 3 SCA  
 4 RECON  
 5 ARTY  
 6 MOVE  
 7 STATUS  
 8 REQUEST  
 9 ABC  
 A FMJ  
 B PREP  
 C NAV  
 D FREE TEXT

SIZE OF HEADER &amp; COMMON BLOCK = 240

Figure 21 Simnet PDU Header &amp; Common



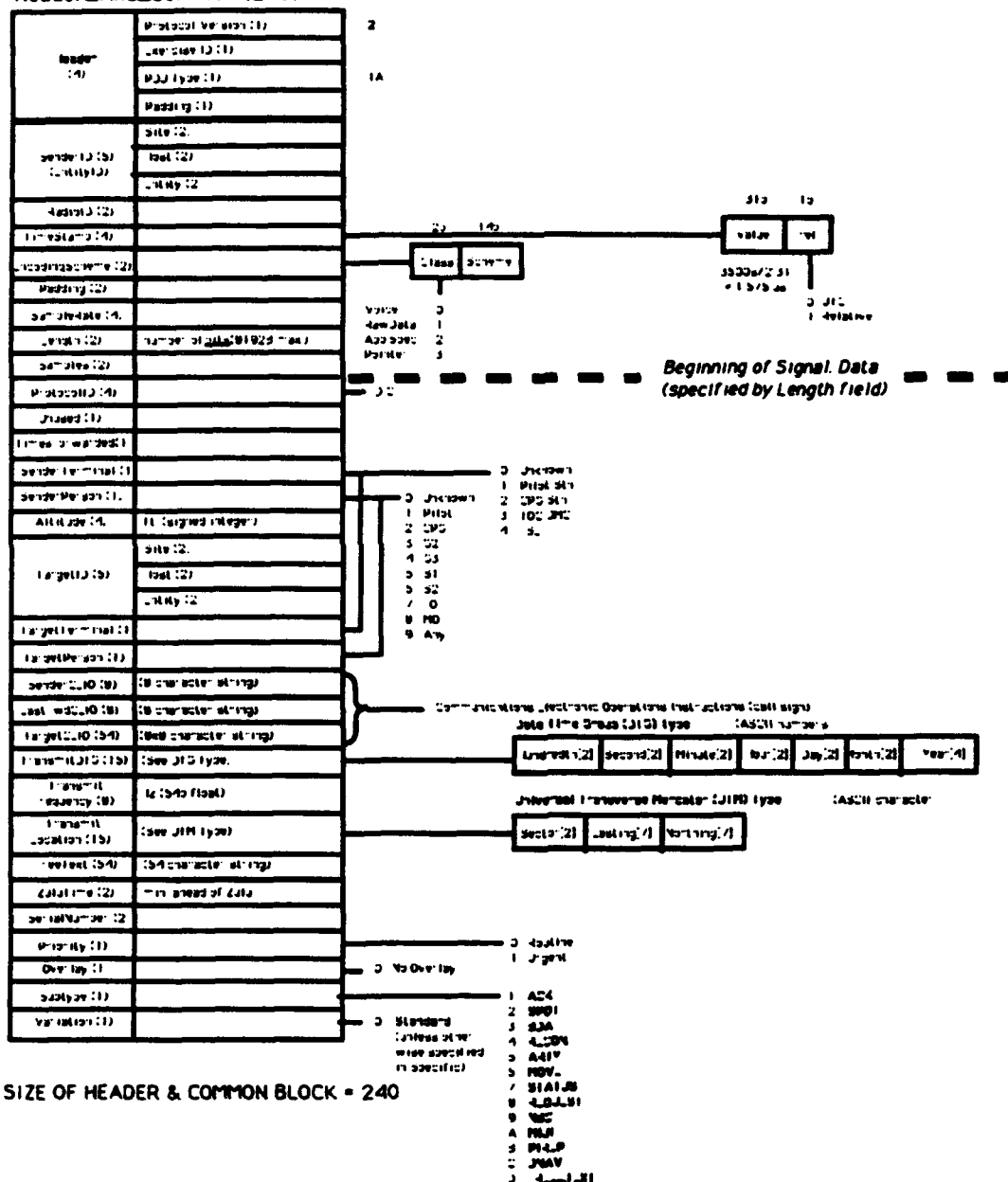
## 40.2 DIS Header and Common Block

1 A DMC (SIGNAL) PDU: DIS

**Rev. Date:**  
**1/6/93**

1. The first part of the document is a list of names and addresses, which appears to be a directory or a list of contacts. The names are written in a stylized, cursive script, and the addresses are listed below them.

**Header\_And\_Common (240):**



### Figure 22 Simnet PDU Header & Common

## 40.3 BDA PDU Specific

Rev. Date:  
12/16/92

03 Subtype = BDA  
00 Variation = Standard

Specific (40):

StrikeTime Start (16)	(See DTG Type)
StrikeTime End (16)	(See DTG Type)
TargetCategory (1)	
TargetType (1)	
Targets Destroyed (2)	
Percent Coverage (1)	
Unused (3)	

0 Not Entered  
1 Unknown  
2 ADA  
3 SAM  
4 Tank  
5 Whld  
6 Trkd  
7 Acft  
8 Trps  
9 Uncl

0 Not Entered  
1 0 %  
2 25 %  
3 50 %  
4 75 %  
5 100 %

LEGEND

B = Bytes (default if not specified)  
b = bits  
all values specified in hex  
fields are unsigned if not specified

TOTAL SIZE = 240 + 40 = 280 Bytes (includes Header & Common Block)

Figure 23 Battle Damage PDU Specific

**40.4 MOVCMO PDU Specific**

D C DMC PDU Specific:

06 Subtype = MOVE

00 Variation = Standard

Specific (48):

TargetID (16)	(16 character string)
Task (1)	
who (1)	
when (1)	
Unused (3)	
Zulu Time (2)	Minutes ahead of Zul
where (8)	(8 Character String)
DTG (16)	(See DTG Type)

0 Not Entered  
 1 MOV TO  
 2 HOLD AT  
 3 CONT MSN  
 4 RENOV AT  
 5 ENGAGE TGT AT  
 6 MOVING TO  
 7 HOLD AT  
 8 ARRIVING AT  
 9 PSNG THRU  
 A DEPARTING FROM

0 Not Entered  
 1 ME  
 2 YOU  
 3 SEE BELOW

LEGEND  
 B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex .  
 fields are unsigned if not specified

0 Not Entered  
 1 Immed  
 2 when Roy  
 3 AMC  
 4 DTG

Rev. Date:  
 3/16/93

TOTAL SIZE = 240 + 48 = 288 Bytes (includes Header & Common Block)

**Figure 24 MOVCMO PDU Specific**

**40.5 MIJI PDU Specific:**

D C DMC PDU Specific:

0 A Subtype = MIJI

00 Variation = Standard

Specific (96):

Rev. Date:

12/15/92

Affected RadioFreq (8)	(64b float)
Programmed Freq (32)	(4x64b float)
KeypadEntry (16)	(16 character string)
JamStartDTG (16)	(See DTG Type)
JamEndDTG (16)	(See DTG Type)
Description (1)	
PercentLost (1)	
Type (1)	
Unused (5)	

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex  
fields are unsigned if not specified

- |               |               |
|---------------|---------------|
| 0 Not Entered | 0 Not Entered |
| 1 Unknown     | 1 Unknown     |
| 2 Meaoning    | 2 Intermit    |
| 3 Intrusion   | 3 Cont        |
| 4 Jamming     | 4 Imitative   |
| 5 Interfere   | 5 Warbling    |
|               | 6 Music       |
|               | 7 Keypad      |

TOTAL SIZE = 240 + 96 = 336 Bytes (includes Header &amp; Common Block)

Figure 25 MIJI PDU Specific

#### 40.6 PIREP PDU Specific:

D C DMC PDU Specific:

0 B Subtype = PIREP

00 Variation = Standard

**Specific (24):**

Visibility ( )		5 Three Mile		
Restrictions ( )		6 Four Mile		
CloudCover ( )		7 Five Mile		
TurbIntensity ( )			0 Not Entered	1 Not Entered
TurbFrequency ( )			2 None	3 Rain
TurbIntensity ( )			4 Snow	5 Fog
TurbFrequency ( )			6 Smoke	7 Drizzle
TurbIntensity ( )			8 Dust	9 Blizzard
TurbFrequency ( )				
icingType ( )		0 Not Entered	2 Trace	4 Cold
windDirection ( )		3 None	5 Moderate	
windSpeed ( )		4 Clear	5 Severe	
OutsideAirTemp (2)	degrees C (signed int)			
Barometer (4)	inches - g (32s float)			
CloudBase (4)	signed int			
CloudTop (4)	signed int			

Rev. Date:  
 12/15/92

2 = Bytes (default if not specified)  
 2 = bits

**Rev. Date:**  
**12/15/92**

**DATA**  
 8 = bytes (default if not specified)  
 2 = bits  
 all values specified in two  
 fields are truncated if not specified

**TOTAL SIZE = 240 + 24 = 264 Bytes (includes Header & Common Block)**

### Figure 26 PIREP PDU Specific

**40.7 RECON GND ROUTE PDU Specific:**

04 Subtype = RECON  
 00 Variation = GND ROUTE  
 Specific (80): Subtype.GndRoute

Rev. Date:  
 12/03/92

**LEGEND**

B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex  
 fields are unsigned if not specified

EnemyActivity (1)			0 Not Entered
Classification Formula (1)		0 Not Entered	1 None
Unused (6)		1 W	2 Stationary
ClassFormInfo (32)	(32 character string)	2 Type	3 Moving
RouteID (32)	(32 character string)	3 MLC	4 Dug In
Leftover (8)		4 OMC	5 Retreating
		5 OB	6 Advancing
		6 Block	7 Attacking
			8 Damaged
			9 Killed
			A Mobile Kill

**Figure 27 Gnd Route Recon PDU Specific**

March 31, 1993

**40.8 RECON AIR ROUTE Specific:**

01 Variation = AIR ROUTE

Specific (80): Subtype.AirRoute

Rev. Date:

12/03/92

EnemyActivity (1)			0 Not Entered
Obstacles (1)		0 Not Entered	1 None
Unused (6)		1 None	2 Stationary
ClassFormInfo (32) (32 character string)		2 Keypad	3 Moving
RouteID (32) (32 character string)		3 ADA	4 Dug In
Leftover (8)		4 Tower Ant	5 Retreating
		5 Wires	6 Advancing
		6 Terr	7 Attacking
		7 Trees	8 Damaged
		8 Blg	9 Killed
		9 Other	A Mobile Kill

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

**Figure 28 Air Route Recon PDU Specific**

**40.9 RECON Bridge Specific:**

02 Variation = BRIDGE

Specific (80): Subtype Bridge

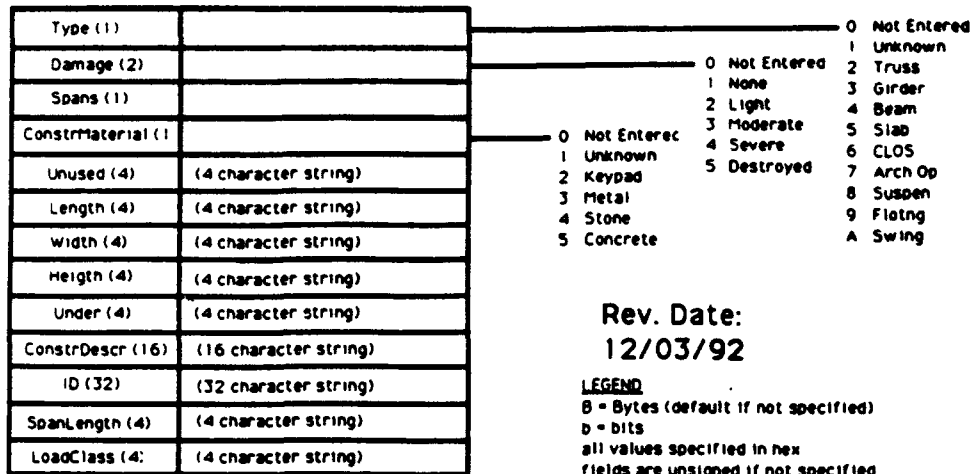


Figure 29 Bridge Recon PDU Specific



**40.10 RECON LZ/PZ PDU Specific**

D C DMC PDU Specific:  
 04 Subtype = RECON  
 03 Variation = LZ\_PZ  
 Specific (80): Subtype.LZ\_PZ

Rev. Date:  
 11/23/92

**LEGEND**

B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex  
 fields are unsigned if not specified

ActivityLikely (1)		
Obstacles (1)		0 Not Entered
Unused (6)		1 None
Obstacles Descr (16)	(16 character string)	2 Keypad
ID (16)	(16 character string)	3 ADA
LZ_Size (16)	(16 character string)	4 Tower Ant
Axis (16)	(16 character string)	5 Wires
Leftover (8)		6 Terr
		7 Trees
		8 Bldg
		9 Other
		0 Not Entered
		1 None
		2 Expected
		3 Possible
		4 Unlikely

**Figure 30 LZ/PZ Recon PDU Specific**

#### 40.11 RECON BP-OP PDU Specific:

04 Variation = BP\_OP

**Specific (80): Subtype.BP\_OP**

EnemyActivity (1)			0 Not Entered
Obstacles (1)		0 Not Entered	1 None
Unused (6)		2 Keypad	2 Stationary
Obstacles Descr (16)	(16 character string;	3 ADA	3 Moving
ID (16)	(16 character string)	4 Tower Ant	4 Dug In
BP_Size (16)	(16 character string)	5 Wires	5 Retreating
Axis (16)	(16 character string)	6 Terr	6 Advancing
Leftover (8)		7 Trees	7 Attacking
		8 Bldg	8 Damaged
		9 Other	9 Killed
			A Mobile Kill

**Rev. Date:**  
**11/23/92**

**LEGEND**

**B** = Bytes (default if not specified)  
**b** = bits  
 all values specified in hex  
 fields are unsigned if not specified

**Figure 31 BP-OP Recon PDU Specific**

**40.12 RECON Crossing PDU Specific:****05 Variation = CROSSING****Specific (80): Subtype.Crossing**

BankSlopeEntry (4):	(4 character string)
BankSlopeExit (4):	(4 character string)
CrossingLength (4):	(4 character string)
CrossingWidth (4):	(4 character string)
CrossingDepth (4):	(4 character string)
CurrentFlow (4):	(4 character string)
ID (8)	(8 character string)
Leftover (32)	

**Rev. Date:****11/23/92****LEGEND****B = Bytes (default if not specified)****b = bits****all values specified in hex****fields are unsigned if not specified****Figure 32 Crossing Recon PDU Specific**

**40.13 ACK PDU Specific**

D C DMC PDU Specific:

01 Subtype = ACK

00 Variation = Standard

Specific (48):

Original SenderID (6)	Site (2)
	Host (2)
	Entity (2)
OriginalSender Terminal (1)	
OriginalSender Person (1)	
AcknowledgeID (6)	Site (2)
	Host (2)
	Entity (2)
Acknowledge Terminal (1)	
Acknowledge Person (1)	
Original SenderCEIO (8)	(8 character string)
Acknowledge CEIO (8)	(8 character string)
OriginalDTG(16)	(See DTG Type)

Rev. Date:

12/03/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

TOTAL SIZE = 240 + 48 = 288 Bytes (includes Header &amp; Common Block)

**Figure 33 Acknowledge PDU Specific**

## 40.14 SPOT PDU Specific

D C DMC PDU Specific:

02 Subtype = SPOT

00 Variation = Standard

Specific (88):

TimeSighted (16)	(See DTG Type)	Rev. Date: 12/03/92	0 Not Entered 1 Other 2 Cont Man 3 Enga 4 Hid 5 Rts Ihold 6 Obsv Tgt	0 Not Entered 1 Unknown 2 ADA 3 SAM 4 Tank 5 whld 6 Trkd 7 Acft 8 Trps 9 Uncl
Observer Location (16)	(See UTM Type)			
TargetQuantity(2)				
Observer Intentions (1)				
TargetType (1)				
TargetActivity (1)				
TargetDirection (1)		<div> <div>MPH</div> <div>Value</div> <div>Speed Type</div> </div> <div>1b 15b</div> <div>LEGEND</div> <div>B = Bytes (default if not specified)</div> <div>b = bits</div> <div>all values specified in hex</div> <div>fields are unsigned if not specified</div>	0 Not Entered 1 N 2 NE 3 E 4 SE 5 S 6 SW 7 W 8 NW 9 Unknown	0 Not Entered 1 None 2 Stationary 3 Moving 4 Dug In 5 Retreating 6 Advancing 7 Attacking 8 Damaged 9 Killed 10 Mobile Kill
TargetSpeed (2)	(See Speed Type)			
Target Location (16)	(See UTM Type)			
TargetUnit (16)	(16 character string)			
Unused (16)				

Figure 34 Acknowledge PDU Specific

#### 40.15 NBC-1 PDU Specific

09 Subtype = NBC

00 Variation = NBC-1

**Specific (64): Subtype.NBC\_1**

[illegible]

**Figure 35 NBC-1 PDU Specific**

#### 40.16 NBC-4 PDU Specific

09 Subtype = NBC

03 Variation = NBC-4

**Specific (64): Subtype.NBC\_2**

Description (1)			
BurstType (1)			0 Not Entered 1 Unusual 2 Initial 3 Increasing 4 Decreasing 5 Read 6 Special 7 Series 8 Verification 9 Summary
DeliveredBy (1)		0 Not Entered 1 Unknown 2 Arty 3 Mortar 4 Rocket 5 Missile 6 Bomb	0 Not Entered 1 None 2 Unknown 3 Surface 4 Air 5 Gnd
CloudHtUnits (1)			
CloudWidth (4)	in degrees (32b float)		
CloudDescr (16)	(16 character string)		
CloudHeight (2)			
DoseRate (2)			
Unused (2)			
Leftover (32)			

Rev. Date: 11/30/92

LEGEND  
 B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex  
 fields are unsigned if not specified

**Figure 36 NBC-4 PDU Specific**

```
09 Subtype = NBC
04 Variation = NBC-5
Specific (64): Subtype.NBC_Negative
```

[illegible]

**LEGEND**  
B = Bytes (default if not specified)  
b = bits  
all values specified in hex  
fields are unsigned if not specified

### Figure 37 NBC-5 PDU Specific



**40.18      ARTILLERY REPEAT PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

00 Variation = REPEAT

Specific (56): Subtype.Repeat

Rev. Date:

11/25/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

**Figure 38      Artillery Repeat PDU Specific****40.19      ARTILLERY CANCEL PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

01 Variation = CANCEL

Specific (56): Subtype.Cancel

Rev. Date:

11/25/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

**Figure 39      Artillery Cancel PDU Specific****40.20      ARTILLERY CHECK PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

02 Variation = CHECK

Specific (56): Subtype.CheckFire

Rev. Date:

11/25/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

**Figure 40      Artillery Check PDU Specific****40.21      ARTILLERY CNO PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

03 Variation = CNO

Specific (56): Subtype.CNO

Rev. Date:

11/25/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

**Figure 41      Artillery CNO PDU Specific**

**40.22 ARTILLERY SHIFT PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

04 Variation = SHIFT

Specific (56): Subtype.Shift

Rev. Date:

11/25/92

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

MissionID (16)	(16 character string)	
TargetID (16)	(16 character string)	
MissionStatus (1)		0 Not Entered
FireforEffect (1)		1 Requested
Unused (7)		2 Ready
Message (16)	(16 character string)	3 Shot
		4 Splash

Figure 42 Artillery Shift PDU Specific

**40.23 ARTILLERY SHOT PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

07 Variation = SHOT

Specific (56): Subtype.Shot

Figure 43 Artillery Shot PDU Specific

**40.24 ARTILLERY SPLASH PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

08 Variation = SPLASH

Specific (56): Subtype.Splash

Rev. Date:

11/25/92

MissionID (16)	(16 character string)
TargetID (16)	(16 character string)
MissionStatus (1)	
RoundsFired (1)	
ImpactTime (1)	in seconds
Unused (5)	
Leftover (16)	

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

- 0 Not Entered
- 1 Requested
- 2 Ready
- 3 Shot
- 4 Splash

**Figure 44 Artillery Splash PDU Specific****40.25 ARTILLERY EOM PDU Specific**

D C DMC PDU Specific:

05 Subtype = ARTY

09 Variation = EOM

Specific (56): Subtype.EndofMission

Rev. Date:

11/25/92

MissionID (16)	(16 character string)
TargetID (16)	(16 character string)
MissionStatus (1)	
Disposition (1)	
Casualties (1)	
RecordTarget (1)	
CasualtyNumber (2)	
Unused (2)	
PointNumber (16)	(16 character string)

**LEGEND**

B = Bytes (default if not specified)

b = bits

all values specified in hex

fields are unsigned if not specified

- 0 Not Entered
- 1 Requested
- 2 Ready
- 3 Shot
- 4 Splash

- 0 Not Entered
- 1 Burning
- 2 CNO

- 0 FALSE
- 1 TRUE

**Figure 45 Artillery EOM PDU Specific**

**40.26 STATUS PDU Specific**

D C DMC PDU Specific:  
 07 Subtype = STATUS  
 00 Variation = Standard

Specific (16):

Rev. Date:  
 11/23/92

Fuel (1)	in lbs
Elements (1)	
FailedEquip (3)	(array of 3)
Hellfires (1)	
Stingers (1)	
Rockets (1)	
Rounds (1)	
RequestType (1)	
Unused (5)	

- 0 Not Entered  
 1 Eng1  
 2 Eng2  
 3 Guns  
 4 Radio  
 5 Dgntr  
 6 Rktpod  
 7 Msl Stn  
 8 Radar  
 9 Laser
- 0 Standard  
 1 Automatic

**LEGEND**

B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex  
 fields are unsigned if not specified

TOTAL SIZE = 240 + 16 = 256 Bytes (includes Header & Common Block)

**Figure 46 Artillery Status PDU Specific**

## 40.27 REQUEST PDU Specific

```
D C  DMC PDU Specific:
08  Subtype   = REQUEST
00  Variation = Standard
```

**Specific (8):**

ReportType (1)		0 Not Entered
ReconType (1)		1 Status
Unused (6)		2 Spot
		3 Recon
		4 Movement
		5 Prep
		6 MJI

**TOTAL SIZE = 240 + 8 = 248 Bytes (includes Header & Common Block)**

### Figure 47 Artillery Request PDU Specific

**40.28 DNAV PDU Specific**

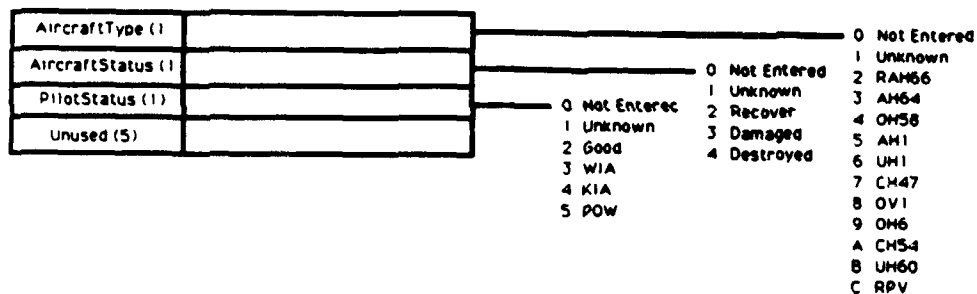
D C DMC PDU Specific:  
 0 C Subtype = DNAV  
 00 Variation = Standard

Rev. Date:  
 11/23/92

**LEGEND**

B = Bytes (default if not specified)  
 b = bits  
 all values specified in hex  
 fields are unsigned if not specified

Specific (8):



TOTAL SIZE = 240 + 8 = 248 Bytes (includes Header & Common Block)

**Figure 48 DNAV PDU Specific**

**40.29 FREE TEXT PDU Specific**

D C DMC PDU Specific:

O D Subtype = FREE TEXT

00 Variation = Standard

Rev. Date:

11/23/92

LEGEND

B = Bytes (default if not specified)

b = Bits

all values specified in hex

fields are unsigned if not specified

**Specific (256):**

FreeText(256)	(256 character string)
---------------	------------------------

TOTAL SIZE = 240 + 256 = 496 Bytes (includes Header &amp; Common Block)

**Figure 49 Free Text PDU Specific**

## **50. Appendix E - DMCC Functional Development Tests.**

The following pages detail the functional development tests which were used to verify proper operation of the DMCC software. These tests fall into several categories as noted here:

### **Developmental Thread Tests**

1. Stand Alone IPC Thread Test
2. Dual Client Stand Alone Thread Test

### **Transmission and Reception Tests**

3. Transmitted PDU Content Accuracy Test
4. PDU Reception Accuracy Text
5. DMCC X-terminal Configuration Test
6. Dual Host, Dual Client SIMNET Compliance Test

### **User Interface Tests:**

7. Window Content Verification Tests
8. Graphical User Interface Window Navigation Test
9. Re-Use and Reply; Acknowledge Tests
10. Miscellaneous Functional Tests

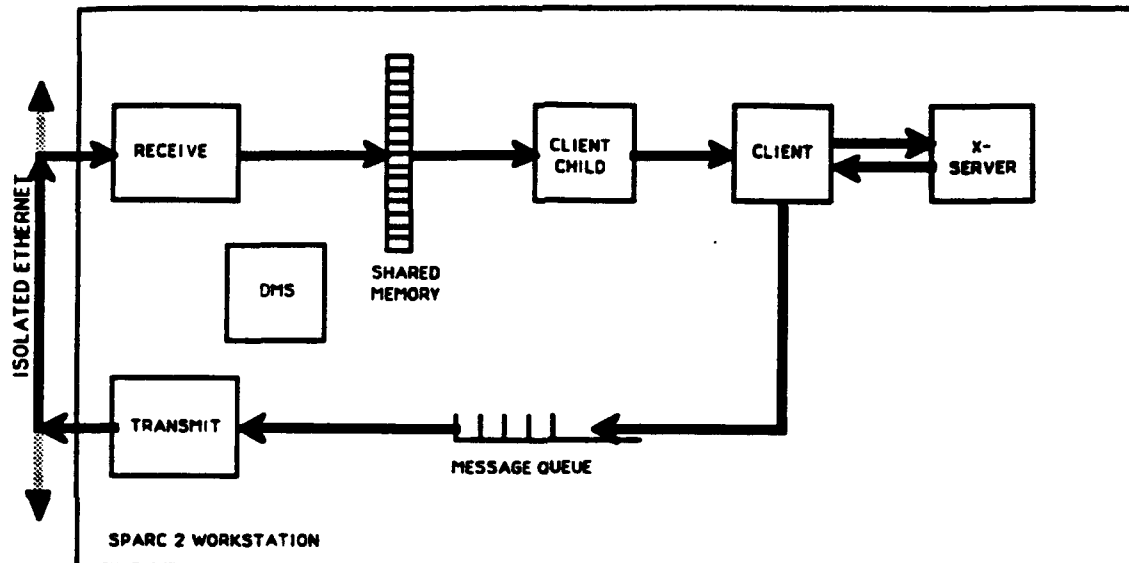


## Developmental Thread Tests

### 50.1. Single Client Stand Alone IPC Thread Test

#### Purpose:

Verify operation of the Unix System V Interprocess Communication (IPC) elements in the DMCC software.



### 1. DMCC SINGLE CLIENT STAND ALONE DEVELOPMENTAL THREAD TEST (FREE TEXT MESSAGE; HARD CODED DESTINATION; NO CLIENT MESSAGE ARCHIVING OR RETRIEVAL)

#### Background:

In the Digital message Communications Console software, a message PDU is created in the client application and transmitted through the outgoing message queue to the concurrent ETHERNET TRANSMIT PROCESS. The TRANSMIT PROCESS then transmits the PDU on the Ethernet. The message is then received by the ETHERNET RECEIVE PROCESS, which asks the MESSAGE SERVER PROCESS where to put the message in shared memory. The RECEIVE process then informs the CLIENT CHILD PROCESS of the message's location through a message queue. The client child forked by the client then informs the client through an unnamed pipe that a new message has arrived for it. The client then retrieves the message from the shared memory and archives it for later retrieval and display.

**Test Description:**

This test will simulate transmission and reception of a Digital Message Protocol Data Unit by one DMCC CLIENT PROCESS. In this developmental test, no archiving of messages by the client is done, and the recipient address is hard-coded into the PDU, instead of being selectable by the user.

**Acceptance Criteria:**

A message must be successfully transmitted and received and the received Message must be identical to the transmitted one.

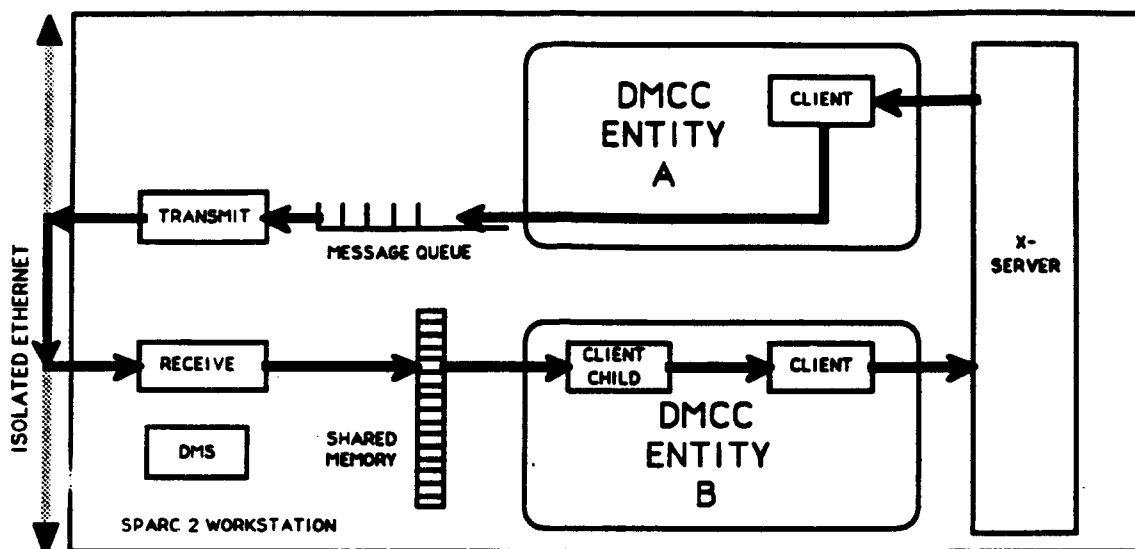
**Test Procedure:**

1. Start the DMCC developmental software using the START DMCC procedure.
2. Select the MSG1 window from the MAIN window.
3. Select the Free Text Message preparation window from the MSG1 window.
4. Enter a unique free text message (a string of 10 or 20 characters). Do not select an addressee for this message.
5. Transmit the message by clicking the mouse pointer on the SEND button.
6. Observe that the message is received back from the Ethernet by the DMCC software.
7. Repeat steps 3 through 6, above, for several different length free text messages: 5 characters, 50 characters, 200 characters, and 255 characters. Allowable characters include \_\_\_\_\_.

## 50.2. Dual Client Stand Alone IPC Thread Test

### Purpose:

Verify proper operation of the DMCC MESSAGE SERVER PROCESS.



# 2.

## DMCC DUAL CLIENT STAND ALONE DEVELOPMENTAL THREAD TEST

(FREE TEXT MESSAGE;  
OPERATOR SPECIFIED DESTINATION;  
FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

### Background:

Each DMCC message contains at least one but no more than eight destinations or addresses. These destinations can take the form of the Call Sign (CEOI) of a destination entity, or a GROUP NAME for a group of DMCC entities to whom the digital message is collectively addressed. The CALL SIGN (CEOI) and GROUP NAMES are eight character alphanumeric identifiers.

However, in this implementation of the DMCC software, the graphical user interface does not allow for the entry of more than one simultaneous address for a particular message, be it a CALL SIGN (CEOI) or a GROUP NAME. The other seven destination fields in the Protocol Data Unit are left blank (filled with zeros).

Each DMCC Entity has a corresponding CALL SIGN (CEOI), and, optionally a list of up to seven GROUP NAMES to which it "claims" membership. More than one client may subscribe to messages in the same group. In this way, every client which subscribes to a group called "TEAM A" will receive messages transmitted to that group (in the same exercise).

During logon of a DMCC entity, the DMCC CLIENT PROCESS requests a logon from the message server. The CLIENT specifies to the message server what its CEOI is and what groups to which it belongs, as well as an EXERCISE ID. The CEOI and the GROUP NAMES are eight character alphanumeric strings. The message server process keeps a separate list of the CEOI, the GROUP NAMES and the Exercise ID for each client

In other words, the message server filters the messages according to destination and exercise.

#### **Test Description:**

This test will evaluate the performance of the message server. This process forms the bridge between the ETHERNET RECEIVE PROCESS and the clients. More than one client application may be running in the same workstation hardware at the same time, each corresponding to exactly one Digital Message Communications Entity. The message server receives a message from the ETHERNET RECEIVE process and sends it to the active client or clients to which it is addressed, and which are operating in the same exercise as the sending DMC entity.

#### **Acceptance Criteria:**

In this test, three DMCC entities (clients) are activated, and tests of sending messages between them are done.

The first test will check to see that a message properly addressed to the target Client and with the proper Exercise ID is correctly handled by the message server and received by the addressee client.

The third test will check to see that a message improperly addressed to the target Client but with the correct Exercise ID is correctly handled by the message server and not passed through shared memory to the client.

The second test will check to see that a message properly addressed to the target Client but with the wrong Exercise ID is correctly handled by the message server and not passed through shared memory to the client.

**Test Procedure:**

1. Start the DMCC software using the START DMCC procedure. Log a client on to the DMS using the logon name "BINKY" and the Exercise ID of 1.
2. Start another client using the START CLIENT procedure. Log this client into the Digital Message Server using the logon name "AKBAR" and an exercise ID of 1.
3. Start a third client using the START CLIENT procedure. Log this client into the DMS using the logon name "BONGO" and the exercise ID of 2.
4. Using the BINKY client, transfer to the ADDRESS LIST EDIT window and add the names "BONGO", "AKBAR" and "JEFF" to the list.
5. Using the AKBAR client, transfer to the ADDRESS LIST EDIT window and add the names "BINKY", "BONGO", AND "JEFF"

**ADDRESS LISTS FOR THE TWO CLIENTS IN THIS TEST**

BINKY CLIENT	AKBAR CLIENT	BONGO CLIENT
BONGO	BINKY	
AKBAR	BONGO	
JEFF	JEFF	

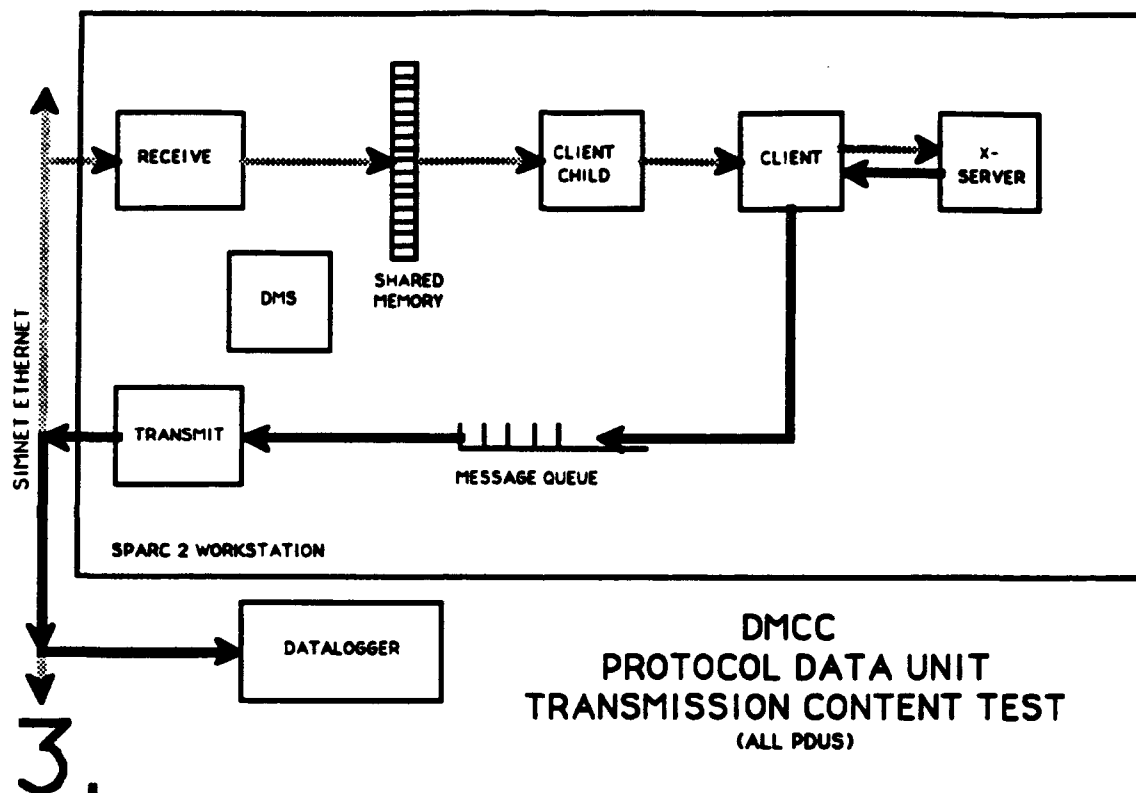
6. Using the BINKY client, transfer to the SPOT report window and send a spot report to the AKBAR client. Using the AKBAR client, transfer to the MSG screen and view the received messages. The spot report from the Binky client should be listed in the mailbox. Select this message and click READ to read it. You should see the spot report you sent from the BINKY client.
7. Using the BINKY client, transfer to the Reports window send a Spot report to the address JEFF.
8. Using the AKBAR client, transfer to the MSG screen and check to see that the second message from the BINKY client was NOT received.
9. Start another client and log on as JEFF, with Exercise ID 2. Go to the address list edit window and enter the address BONGO.
10. Using the JEFF client in Exercise 2, send a spot report to BONGO.
11. Transfer to the BONGO client and check to see that the message from JEFF was NOT received (since JEFF is in Exercise 2, and BONGO is in Exercise 1.)

## DMCC Transmission and Reception Tests

### 50.3. Transmitted PDU Content Accuracy Test.

#### Purpose:

Verify correct transmission of message PDUs.



#### Background:

In this implementation, the Digital Message Communications Console is capable of sending seven different kinds of messages.

#### Test Description:

In this test, each of the different kinds of messages will be transmitted by the DMCC onto the network and logged by the Datalogger for manual inspection.

**Acceptance Criteria:**

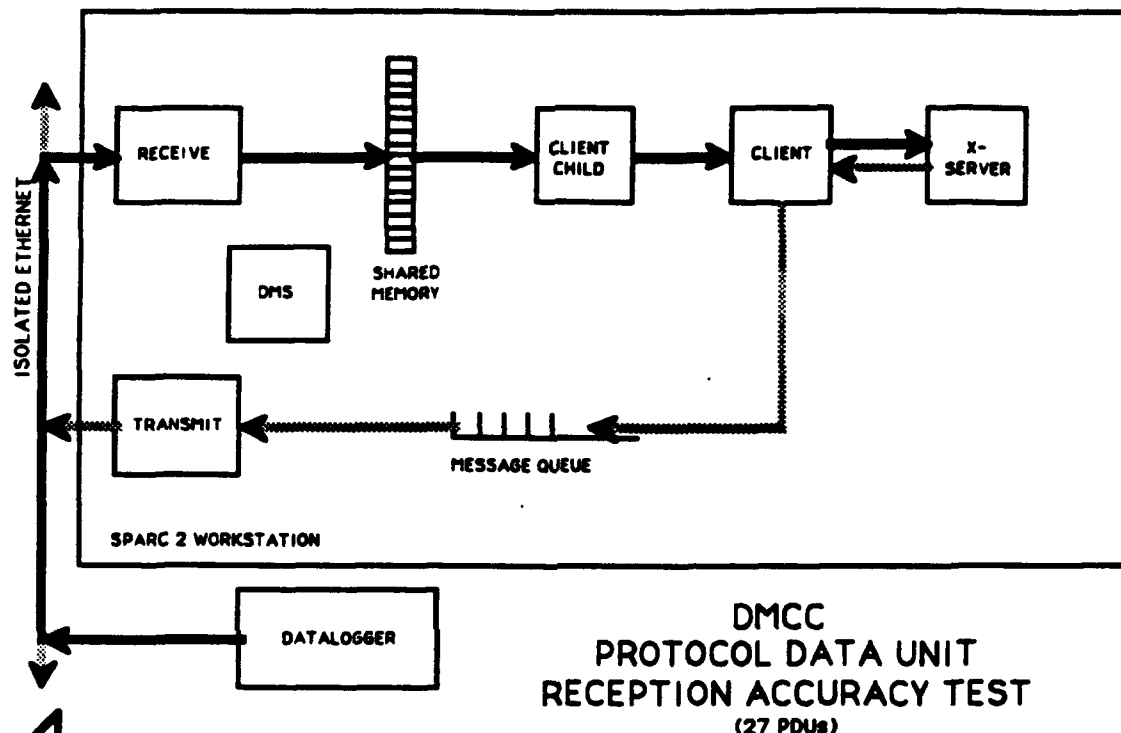
Each Message PDU will be examined for content correctness.

**Test Procedure:**

1. Start the DMCC software using the DMCC start procedure.
2. Start a Client, and log on to the Digital Message Server as REAGAN, with Exercise ID 1.
3. Start the Datalogger and set it to capture Digital Message PDUs only.
4. Using the REAGAN client, send a spot report to the datalogger.
5. Send a MTO report to the datalogger.
6. Send a SHOT report to the datalogger.
7. Send a SPLASH report to the datalogger.
8. Send a REQUEST report to the datalogger.
9. Send a MOVCMD report to the datalogger.
10. Send a FREE TEXT report to the datalogger.
11. Stop the datalogger.
12. Print out the PDUs which were received by the datalogger from the REAGAN client.
13. Inspect the PDUs received from the REAGAN DMCC client for content correctness. You should see the exact fields which you sent from the DMCC for each message type.

#### 50.4. DMCC PDU Reception Accuracy Test.

**Purpose:** To verify that the DMCC software properly decodes PDUs it receives from the network.



#### Background:

In this implementation, the DMCC software can receive, record, retrieve and display 28 different kinds of messages, most of which are actually incapable of being transmitted by the DMCC itself. These are messages which will (maybe) be transmitted by the simulated Comanche helicopter mission equipment package software, and represent digital communications between the helicopter aircrew and ground force elements and command centers.

#### Test Description:

For this test, the PDU Builder will be used to manufacture digital message PDUs which will be stored in a file. These will then be transmitted onto the network by the Anna Tool for reception, storage, and retrieval by a Digital Message Communications Console. The Message Read function of the DMCC will be



used to examine the contents of the received message, and these contents will be checked against the PDU.

#### **Acceptance Criteria:**

Each phony PDU will be constructed ahead of time and the contents noted. These PDUs will be contained in datalogger PDU files which can be read by the Datalogger and transmitted on the net. As each PDU is sent by the Datalogger and received by the DMCC, the PDU contents are checked against the displayed information. The test will be considered to be passed when each type of message is correctly received and displayed by the DMCC software.

#### **Test Procedure:**

1. Using the PDU Builder, prepare a set of PDU files for each PDU type in the Digital Message Protocol. Vary the fields of data between each of the PDUs of a particular type. Prepare enough PDUs to fully test each enumerated type in each kind of PDU. For example, in the MOVCMMD PDU, the TASK enumerated type field contains 11 selections: Not Entered, To, Hold Cont Ms, Rendez At, Enga Tgt, Moving to, Hldg At, Arrivg At, Passg Thru, and Departg From. The WHO enumerated type field contains six selections, and the WHEN field contains five. For this example, 11 PDUs would need to be prepared, since 11 is the number of selections in the task with the most selections. Each PDU would have one of the 11 selections for the TASK field. The selections for the WHO and WHEN fields are to be varied as well, but some of the selections will have to be used more than once since there are only 5 selections and 11 different PDUs.

Build these PDUs with the DESTINATION field set to MADONNA, and the exercise ID set to 1.

Build some of the PDUs of each type with priority set to URGENT and some set to ROUTINE.

Build these PDUs with the date time group set to a constant date: 0705 March 3, 1993.

2. Build a set of 12 SPLASH PDUs to test the Date Time Group feature, by varying the date field to include each month. Vary the time portion of the field to allow tests of 0000 hours, 0600, 1159 and 2359 hours. Vary the seconds field. Obviously, an exhaustive test of all combinations of time is not practical; the attempt here is to just verify that the date and time are going thru for a representative selection of the possible cases.

3. Using the PDU Viewer, print the contents of each PDU out on paper.

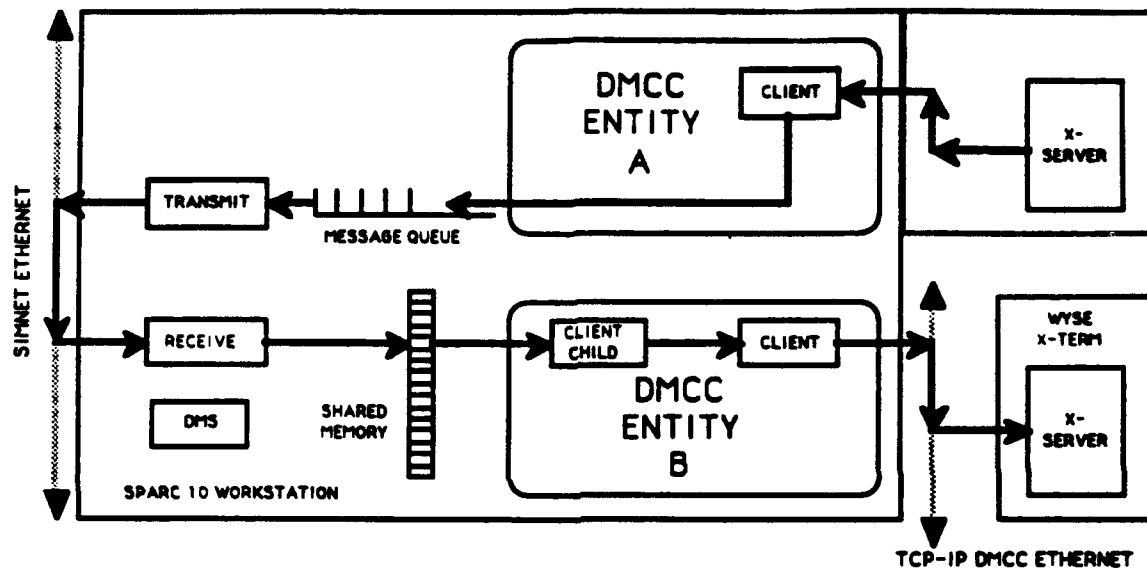
4. Connect the DMCC to SIMNET.

5. Start the DMCC software using the DMCC Start Procedure.
6. Log on a CLIENT to the digital message server under the name MADONNA, using Exercise ID 1.
7. Set the Datalogger to log DMC PDUs.
8. Using the Anna Tool, transmit 15 PDUs into the DMCC.
9. Using the DMCC, transfer to the MSG1 screen, select a PDU from the Message List and click READ.
10. Check the contents of the message as shown on the DMCC MSG READ window against the known contents of the PDU from the PDU Viewer. For all PDUs, you should notice no discrepancies between the message contents and the PDU printout.

## 50.5. DMCC X-terminal Configuration Test

### Purpose:

This test confirms the DMCC operates properly in a dual client, single platform environment. It involves two client processes, one whose x-server resides in the workstation, and one whose x-server resides in a X-terminal.



# 5.

## DMCC X-TERMINAL CONFIGURATION TEST

(7 MESSAGE TYPES • ACKNOWLEDGE;  
OPERATOR SPECIFIED DESTINATION;  
FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

### Background:

The Digital Message Communications Console software operates in a client/server X environment. Individual client applications interface with the network through the Digital Message Server and the network interface software. More than one DMCC client application may be simultaneously running in one workstation. Individual DMCC application windows may be displayed on the console display of the workstation or on X-terminals connected through a normal TCP-IP Ethernet to the workstation.

### Test Description:

This test verifies operation of two client applications in the workstation at the same time, one whose server is resident in the workstation and one whose server is resident in a Wyse X-terminal.

**Acceptance Criteria:**

This test is considered to be passed if proper operation of the send and receive functions is observed for each of the two DMCC client entities.

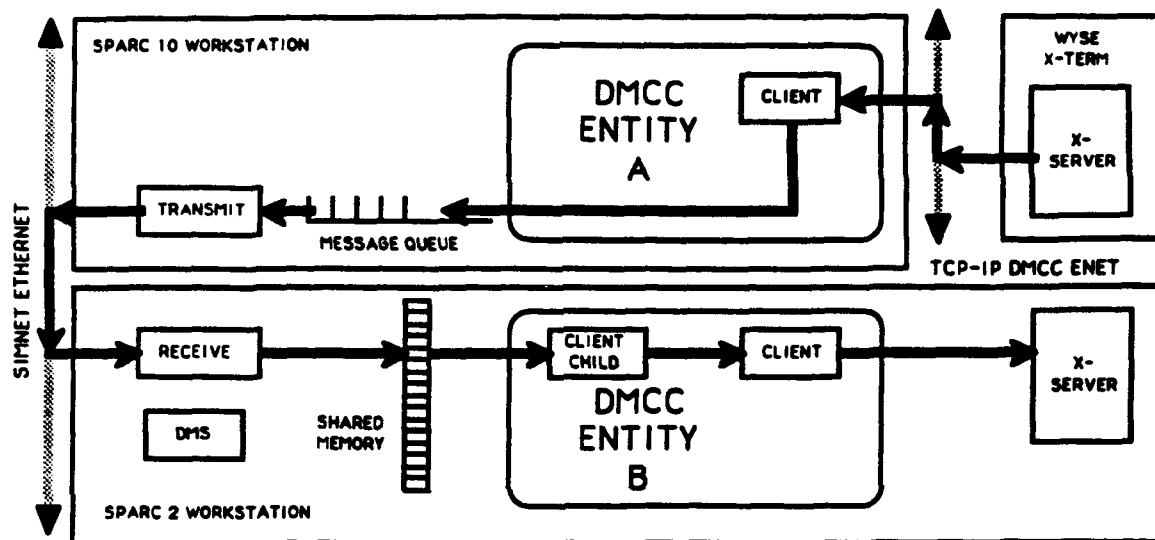
**Test Procedure:**

1. Start the DMCC software in the workstation using the DMCC start up procedure.
2. At the console of the Workstation, log a client on to the Digital Message Server using the DMCC call sign SEAHAWK, and Exercise ID 1.
3. At the X-terminal, log a DMCC client onto the Digital Message Server using the call sign NATE and Exercise ID 1.
4. Using the X-term client, NATE, send some digital messages to the client SEAHAWK. Verify that they are received properly by SEAHAWK.
5. Using the Workstation client, SEAHAWK, send some various digital messages to NATE. Verify that they are received properly by NATE.

## 50.6. Dual Host Dual Client SIMNET Compliance Test

### Purpose:

The purpose of this test is to verify operation of the DMCC software in the context of separate host platforms.



6.

### DMCC DUAL HOST DUAL CLIENT SIMNET COMPLIANCE TEST

(SELECTED MESSAGE TYPES;  
OPERATOR SPECIFIED DESTINATION;  
FULL CLIENT MESSAGE ARCHIVING & RETRIEVAL)

### Background:

The DMCC is intended to be able to provide simulated digital communications between message entities (client applications) hosted on one machine or on several machines connected to the SIMNET ETHERNET or to the Distributed Interactive Simulation network via the protocol gateway.

### Test Description:

In this test, two host platforms are used to test message interoperability on the actual SIMNET network. The Datalogger is used to capture the session for playback.

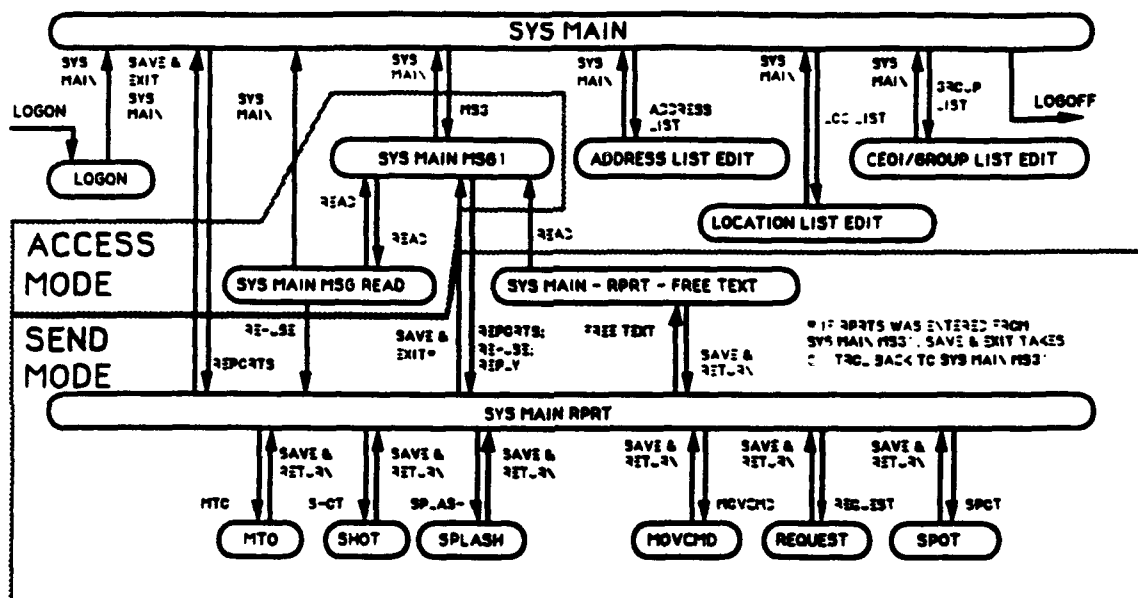
## User Interface Tests

### 50.7. Window Content Verification Test

#### Purpose:

The purpose of this test is to verify that each window in the DMCC client application contains the widgets it needs to do its job.

### DMCC MENU TOPOLOGY



#### Background:

The Digital Message Communications Console software uses a OSF/Motif Graphical User Interface running in a X-Windows windowing environment under SunOS, Sun Microsystems UNIX operating system. The Graphical User Interface consists of a number of Windows which contain buttons for clicking with the mouse, areas for entry of text, lists of names, addresses and locations, and graphical objects designed to simulate to some extent the appearance of the Communications Subsystem of the RAH-66 Comanche Helicopter Mission Equipment Package. These objects are called WIDGETS in the vernacular of the OSF/Motif Graphical User Interface.

#### Acceptance Criteria:

This test shall be deemed to have been passed if all required Widgets are present in all of the DMCC GUI windows.

**Test Description:**

This test simply checks each of the 13 screens in the DMCC GUI for the correct widget content.

**Note:**

It should be noted that there do exist graphical icons and features in some windows which do not have functionality in the context of the Digital Message Communications Console software. These are provided to increase the level of fidelity of the simulation of the Comanche Mission Equipment Package. Those objects are not included in the Window Widget List in appendix WW.

**Test Procedure:**

1. Start the DMCC software using the DMCC start procedure.
2. Examine the logon screen for the proper widget content. The widget content of the logon screen and the other screens in the DMCC is found in the DMCC Graphical User Interface Widget Content List, which is an appendix to this document. Check that each widget identified in the Widget List is present in this window.
3. Log a client on to the Message Server under the client name HOMER and the exercise ID 1.
4. Examine the Sys Main screen for widget content.
5. Transfer to the CEOI/Group List window and examine it for widget content.
6. Transfer back to the Sys Main window then to the Location List Window. Examine the Location List Window for widget content.
7. Transfer back to the Sys Main Window and then transfer to the Address List Window. Examine the Address list window for widget content. Enter HOMER and click the ADD button.
8. Transfer back to the Sys Main Window and then transfer to the MSGS window. Examine the MSGS window for widget content.
9. Transfer to the Reports window and examine it for widget content.
10. Transfer to the MTO window and examine it for widget content.

11. Transfer back to the REPORTS window and then to the SHOT window. Examine the SHOT window for widget content.
12. Transfer back to the REPORTS window and then to the SPLASH window. Examine the SPLASH window for widget content.
13. Transfer back to the REPORTS window and then to the MOVCMMD window. Examine the MOVCMMD window for widget content.
14. Transfer back to the REPORTS window and then to the REQUEST window. Examine the REQUEST window for widget content.
15. Transfer back to the REPROTS window and then to the SPOT window. Examine the SPOT window for widget content.
16. Transfer back to the REPORTS window and then to the FREE TEXT window. Examine the FREE TEXT window for widget content.
17. Enter the a free text message, select HOMER as the addressee, and click SEND. This will cause the client to send the free text message to itself, so you can select it for viewing.
18. Wait for 5 seconds.
19. Click the READ button to get back to the MSG window. Select the message which was just received, and press click READ again.
20. Examine the READ window for widget content.
21. Click SYS MAIN to get back to the SYS MAIN window.
22. Click LOGOFF to Log off.



## **50.8. Graphical User Interface Window Navigation Test**

### **Purpose**

The purpose of this test is to ensure that all window navigation pathways exist and are operational.

### **Background**

The DMCC uses a X-Windowing system, with various X-Windows representing various "screens" or "menus" in the Comanche Mission Equipment Package. Control is passed from one window to another upon certain actions by the user.

### **Acceptance Criteria**

Acceptance criteria for this test are that all window navigation pathways exist and are operational.

### **Test Description**

This test involves starting a DMCC client process, logging it on to the Server, and moving about the window structure to verify each pathway.

### **Test Procedure**

Start the DMCC using the DMCC Start Procedure. Logon a client as MARGE. The following paragraphs describe pathways between windows. Check that each pathway exists and is operational.

Click ADDRESS LIST and check that control passes to the ADDRESS LIST EDIT MENU.

Click SYS MAIN and verify that control passes to the SYS MAIN MENU window.

Click LOC LIST and verify that control passes to the LOCATION LIST EDIT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click CEOI/GROUP LIST and verify that control passes to the CEOI/GROUP LIST EDIT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to the SHOT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to the SHOT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click MOVCMMD and verify that control passes tot the MOVCMMD window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click MOVCMMD and verify that control passes to the MOVCMMD window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click SAVE & RETURN and verify that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click CLEAR & RETURN and verify that control passes to the RPRTS window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click REPORTS and verify that control passes to the RPRTS window.

Click MTO and verify that control passes to the MTO window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SHOT and verify that control passes to me SHOT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SPLASH and verify that control passes to the SPLASH window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click MOVCMO and verify that control passes to the MOVCMO window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click REQUEST and verify that control passes to the REQUEST window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click SPOT and verify that control passes to the SPOT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

Click MSGS and verify that control passes to the MSGS window.

Click RPRTS and verify that control passes to the RPRTS window.

Click FREE TEXT and verify that control passes to the FREE TEXT window.

Click SYS MAIN and verify that control passes to the SYS MAIN window.

## **50.9. Re-use and Reply; Acknowledge Tests**

### **Purpose:**

Verify proper operation of the Reuse (forwarding) and Reply functions in the DMCC Client.

### **Background:**

In the Digital Message Communications System, a mechanism is provided to allow messages which are received to be forwarded to other DMCC entities. This is done by allowing the operator to exit the MESSAGES screen directly to the REPORTS menu, where he can prepare another message to accompany the forwarded message as it is sent on.

The DMCC also allows the user to REPLY to a message directly without having to select the addressee.

The DMCC software is sends an ACKNOWLEDGE PDU to the sender of messages which are read by the user. This acknowledge PDU causes an ACKNOWLEDGE pop-up window to be displayed on the sender's terminal.

### **Test Description:**

This three client test will demonstrate proper operation of the Re-use (Forward) function by sending a message from one client to a second client, and then having the second client forward that message, along with another message, to a third client.

Several REPLY messages will be sent as well.

Checks for proper transmission and reception of Acknowledge PDUs will be undertaken as well.

### **Acceptance Criteria:**

This test will be deemed to have been passed if messages are properly forwarded and replied to, and if ACKNOWLEDGE messages are sent and received correctly.

### **Test Procedure:**

1. Start the DMCC software using the START DMCC procedure.

2. Log on to the Server as TOC.
3. In the TOC client, enter the ADDRESS LIST EDIT window and add the CEOI "LEADER" to the list. Return to the SYS MAIN window.
4. Start a second client and log it on to the server as SANTA. Add the names TOC and WINGMAN to LEADER's address list.
5. Start a third client named WINGMAN.
6. Go back to the TOC client and click MSGS to enter the Messages window.
7. Click RPRTS to enter the REPORTS window.
8. Click FREE TEXT to enter the FREE TEXT window.
9. Enter the following free text message:

**PROCEED WAYPT 4 AWAIT BEGINNING OF HF JAMMING.**

10. Send the message to LEADER.
11. Now go to the LEADER client and access the message sent by the TOC client.
12. Dismiss the message window. Make sure the message is still highlighted in the mailbox.
13. Click REPLY.
14. In the REPORTS screen, select FREE TEXT .
15. Go back to the TOC client and see if an acknowledge PDU was received from LEADER.
16. Go to the LEADER client and enter the following FREE TEXT message:  
**WILL COMPLY. WILL AWAIT FRIENDLY MIJI ACTIVITY**
16. Click Send Routine.
17. Go to the TOC client and verify that the reply message has arrived.
18. Go to the mailbox and read the reply message.
19. Dismiss the reply message.

20. Click **REPLY**, and go to the **FREE TEXT REPORT** screen.

21. Enter the following free text message:

**IGNORE MEACONING ON 1022 MHZ**

22. Click **SEND URGENT**.

23. Go to the **LEADER** client and access the message from **TOC**.

24. Go to **TOC** and see if an **ACK** was received from **LEADER**.

25. Go to **LEADER**. Read the message, then dismiss it.

25. Click **REPLY**

26. Go to free text and enter the following message:

**WILL AWAIT JAMMING; IGNORE MEACONING**

27. Click **Send Routine**.

28. The message from **TOC** should still be highlighted in the mailbox.

29. Click **RE-USE**.

30. Go to **Free Text**.

31. Enter the following **Free Text Message**:

**WINGMAN, PLEASE SEE ENCLOSED AND COMMENT ASAP. STATUS OF RADIOS. JAMMING TO BEGIN 1125.**

32. Click **SEND URGENT**.

33. Go to the **WINGMAN** client and see if the messages arrived. There should be the original message from the **TOC** client, plus the new one from **LEADER**.

34. Access and read the messages.

35. Go back to **LEADER** and see if **LEADER** got **ACKS** back for **EACH** of the two messages.

## **50.10. Miscellaneous Functional Tests**

### **Purpose:**

The purpose of this test is to verify miscellaneous operational features of the Digital Message Communications Console software. These include:

- Message Deletion
- Client Death
- Message Prioritization
- Operator errors entering time
- Operator errors entering UTMs
- Escape key handling
- Control key handling
- Reception of messages during report preparation
- Reception of messages while an alert box is active
- Case insensitivity of target CEOI/Group Names
- Default Addressing for Reply
- Empty Message Send Handling

## **50.11. Functional Stress Tests**

- Message Server Message Queue Overload
- Shared Memory Overload
- Client Message Queue Overload
- High Fan Out Messages
- Free Text String Overload
- Free Text Annotation Overload
- Eight Client Test
- Alternate Group Name Message Routing



## **50.12 Functional Test... Window Widgets**

### **DMCC Window Widget Content List**

All DMCC windows contain the Free Text field, simulate Cockpit Interactive Keyboard buttons (left arrow, right arrow, DEL and CLEAR ALL. The following paragraphs detail the window widgets in addition to these, for each window type.

#### **LOGON WINDOW:**

- LOGON TO NETWORK button
- STAND ALONE button
- LOGON NAME Text Field
- EXERCISE ID Text Field

#### **SYS MAIN WINDOW**

- ADDRESS LIST button
- LOCATION LIST button
- CEOI/GROUP LIST BUTTON
- LOGOUT BUTTON
- MSGs BUTTON
- RPRT BUTTON
- SYS MAIN BUTTON

#### **ADDRESS LIST EDIT WINDOW**

- Add button
- Delete button
- Save & Exit button
- Sys Main button
- UP button
- DOWN button
- New Address Text Field
- Address List

#### **LOCATION LIST EDIT WINDOW**

- ADD button
- DELETE button

- SAVE & EXIT button
- SYS MAIN button
- UP button
- DOWN button
- New Location Text Field
- Location List

#### CEOI/GROUP LIST EDIT WINDOW

- ADD button
- DELETE button
- SAVE & EXIT button
- SYS MAIN button
- UP button
- DOWN button
- NEW CALL SIGN or GROUP NAME text field
- Group List

#### SYS MAIN MSG1 WINDOW

- Message Queue List
- PREV button
- NEXT button
- RPRT button
- READ button
- DELETE button
- RE USE button
- REUSE & INCLUDE button
- REPLY button
- Envelope Icon
- Envelope Icon Number
- SAVE & EXIT button

#### SYS MAIN MSG READ WINDOW

- READ button
- DELETE button
- SYS MAIN button
- NEW MESSAGE display

#### SYS MAIN REPORTS WINDOW

- SPOT button
- MTO button
- SHOT button
- SPLASH button

FREE TXT button  
REQT button  
MOVCMD button  
SYS MAIN button

#### MTO WINDOW

ADRS selection ribbon & button  
REQ ADJ button  
EAT button  
EOM button  
SAVE & RETURN button  
CLEAR & RETURN button  
SND \* ROUT button  
SND \* URG button

\* Note: for the MTO window, the SND ROUT and SND URG buttons will contain the text of what kind of MTO message has been selected for transmission. For example, if the REQ ADJ button has been selected, the buttons will say SND ROUT REQ ADJ and SND URG REQ ADJ.

#### SHOT WINDOW

ADRS selection ribbon & button  
FREE TEXT annotation field  
SAVE & RETURN button  
CLEAR & RETURN button  
SND ROUT button  
SND URG button  
SYS MAIN button

#### SPLASH WINDOW

ADRS selection ribbon & button  
FREE TEXT annotation field  
SAVE & RETURN button  
CLEAR & RETURN button  
SND ROUT button  
SND URG button  
SYS MAIN button

## MOVCMD WINDOW

TASK selection ribbon & button  
WHEN selection ribbon & button  
LCTN selection ribbon & button  
ADRS selection ribbon & button  
FREE TEXT annotation field  
SAVE & RETURN button  
CLEAR & RETURN button  
SND ROUT button  
SND URG button  
SYS MAIN button

## REQUEST WINDOW

TYPE selection ribbon & button  
RECON TYPE selection ribbon & button  
ADRS selection ribbon & button  
FREE TEXT annotation field  
SAVE & RETURN button  
CLEAR & RETURN button  
SND ROUT button  
SND URG button  
SYS MAIN button

## SPOT WINDOW

ENEMY TYPE selection ribbon  
ENEMY TYPE button  
ENEMY ACTIVITY selection ribbon  
DIREC selection ribbon  
DIRC button  
OBS INT selection field  
OBS INT button  
SPEED text entry field  
NUM text entry field  
UNIT text entry field  
FREE TEXT ANNOTATION field  
ADRS selection ribbon & button  
SAVE & RETURN button  
CLEAR & RETURN button  
SND ROUT button  
SND URG button  
SYS MAIN button

**FREE TEXT WINDOW.**

**FULL FREE TEXT field**  
**FREE TEXT ANNOTATION field**  
**ADRS selection ribbon & button**  
**SAVE & RETURN button**  
**CLEAR & RETURN button**  
**SND ROUT button**  
**SND URG button**  
**SYS MAIN button**

**REUSE WINDOW**

**ADRS selection ribbon & button**  
**FREE TEXT annotation field**  
**SAVE & RETURN button**  
**CLEAR & RETURN button**  
**SND ROUT button**  
**SND URG button**  
**SYS MAIN button**

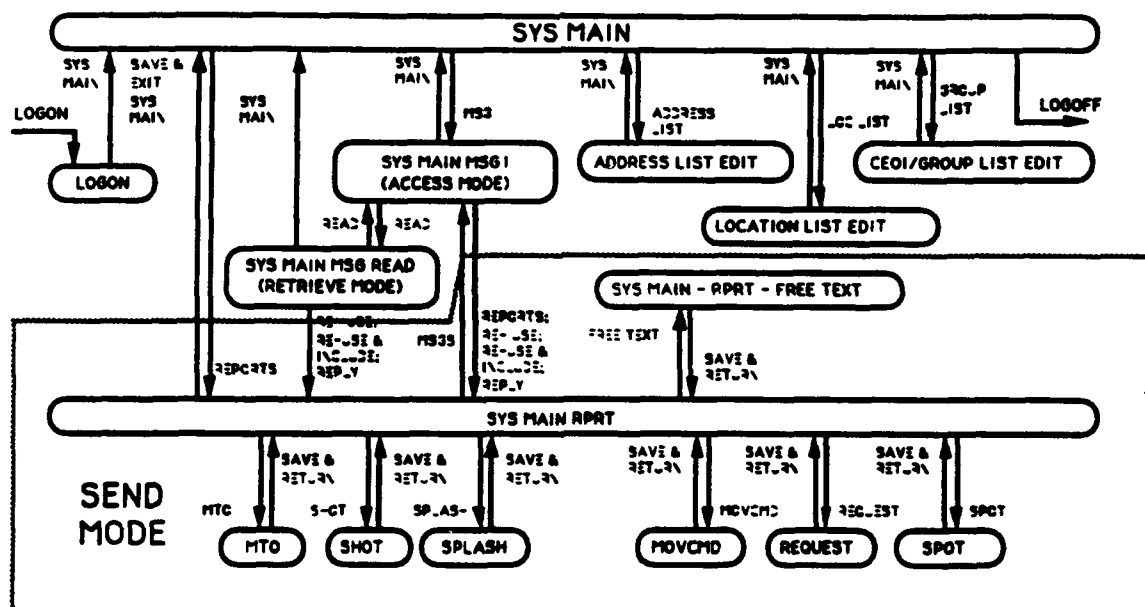
## 60. Appendix F - DMCC Graphical User Interface.

### OPERATIONAL SPECIFICATION FOR THE AIRNET DMCC GRAPHICAL USER INTERFACE (GUI)

#### 60.1 Purpose and Scope:

The AIRNET Digital Message Communications Console uses a OSF/Motif<sup>®</sup> Graphical User Interface running in an X-Windows/UNIX environment on a Sun Microsystems dedicated workstation or client-server X-terminal platform to closely mimic the operation of the messaging subsystem of the Mission Equipment Package on the RAH-66 Comanche reconnaissance/attack helicopter. This interface uses Motif Widgets and icons to effect the simulation of displays and buttons on the System Management Display (SMD), Touchscreen Menu Items (TMI), and Cockpit Interactive Keyboard (CIK) of the Mission Equipment Package. A mouse input device is used to manipulate a mouse pointer on the DMCC workstation or X-terminal display, and to simulate depressing cockpit switches through "clicking" on the screen icons which represent those switches. This document specifies the operation of this Graphical User Interface in the context of the overall DMCC software system specification.

### DMCC WINDOW TOPOLOGY



In this document, Displays, Switches and Icons are identified in THIS COURIER FONT.

## 60.2 General Description of Operation:

### 60.2.1 Interface Elements

The accessing of incoming messages and the control of outgoing messages on the AIRNET Digital Message Communications Console (DMCC) is accomplished by depressing simulated cockpit Touchscreen Menu Items (TMI), activation of Soft Bezel Switches (SBSs) on the simulated System Management Display (SMD), and by entering information in the simulated Cockpit Interactive Keyboard (CIK). The limited CIK will be simulated by using the workstation keyboard with a small window on the DMCC Console Display simulating the CIK display. The X-terminal QWERTY keyboard will be used for entry of textual information.

### 60.2.2 Menu Windows

Fourteen menus will be used by the DMCC GUI:

<b>LOGON</b>
<b>SYS MAIN</b>
<b>ADDRESS LIST EDIT</b>
<b>CEOI/GROUP LIST EDIT</b>
<b>MSG1</b>
<b>MSG READ</b>
<b>RPRT</b>
<b>FREE TEXT</b>
<b>SPOT</b>
<b>REQUEST</b>
<b>MOVCMD</b>
<b>SPLASH</b>
<b>SHOT</b>
<b>MTO</b>

### **60.3 DMCC Incoming Message Control**

#### **60.3.1 Overview**

The DMCC provides a EMAIL-like interface for the listing, access, forwarding, and acknowledgment of incoming Digital Message Communications. Control of the incoming messages is accomplished "in" a number of screen menus which are described in the sections covering the **SYS MAIN** and **SYS MAIN MSG** menu pages.

#### **60.3.2 Call Sign**

Each Digital Message/Communications Entity is given a call sign by its operator.

#### **60.3.3 Groups**

Each entity may optionally specify that it belongs to up to 7 entity groups.

#### **60.3.4 Addresses**

Each Message includes a list of DMC entities (up to 8 entities, identified by their call signs, and/or groups, identified by their group names) which are the addressees or target recipient entities or groups.

#### **60.3.5 Reception**

Each DMC entity can receive only those messages transmitted to it; i.e., messages which are not addressed to its call sign or to a group of which it is a member are ignored by the entity. No alert or acknowledgment of such messages occurs.

#### **60.3.6 TOC - to - TOC Communications**

Thus, for TOC to TOC communications, the TOC which wants to transmit a message to the other TOC must include that TOC's call sign (or the name of a group of which the target TOC is a member) in the list of target recipients.



### **60.3.7 Message Reception**

When a digital message arrives at a DMC entity which is addressed to that entity or is addressed to a group of which that entity claims membership, then that message is received and stored and an alert box is displayed telling the origin of the message and its type (see Sys Main, below).

### **60.3.8 Sender ID & Time Stamp**

Each transmitted message is marked with the CEOI (Computer Electronic Operations Instructions) or CALL SIGN of the transmitting entity and the Date Time Group which tells the time of message transmission.

### **60.3.9 Acknowledge**

Upon SELECTION of a message, the DMCC will transmit an Acknowledge message to the sender, which includes the DTG of the message. Acknowledge messages themselves are not acknowledged.

### **60.3.10 The Alert Box**

An ALERT BOX will appear on the transmitter's screen which says "MESSAGE ACKNOWLEDGED BY <CEOI>". Note that the acknowledger ID is not the GROUP to which the message was sent, but the individual CALL SIGN of the acknowledger. This alert box can then be dismissed by an OK button.

### **60.3.11 Message Forwarding (RE\_USE):**

When a message is selected from the MAILBOX, a RE-USE TMI will become active. This switch will allow the operator to transfer to the REPORTS menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the forward recipient (the person the message is being forwarded to) of the original message, along with the original message.

**60.3.12      Reply**

When a message is selected from the **MAILBOX**, a **REPLY TMI** will become active. This switch will allow the operator to transfer to the **REPORTS** menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the sender of the original message. The **ADRS** bezel will already have selected the sender of the first message as the addressee of the new message to be sent back.

## **60.4 DMCC Outgoing Message Control (REPORTS)**

### **60.4.1 Available Reports**

Available reports shall be presented and accessed via the soft bezel switches. For all types of reports transmitted by the DMCC, the DMCC shall append information to the report selected by the operator. These items include: the sender's identity, the Date Time Group (DTG) telling the date and time the report was sent, and where the sending DMCC was when the report was sent (TOC, FSE, or BATTLEMASTER). Although these pieces of information shall not be seen by the operator of the DMCC, they shall appear on the report received by the addressee individual or group.

### **60.4.2 Reports Function**

The REPORTS function is the means by which the DMCC operator constructs and sends messages out to other DMC entities. The MESSAGES function handles incoming digital message traffic and was discussed above.

### **60.4.3 Accessing the Reports Window**

The standard way of entering the SYS MAIN RPRT menu is by using the REPORTS soft bezel. SYS MAIN RPRT may be exited by pressing the SAVE & EXIT TMI. There are pre-formatted reports, plus a free text report and a "REQUEST REPORT".

### **60.4.4 Menu Wundows**

The following pages describe in detail the operation of each of the MENUS which can appear in the shell window widget resembling the RAH-66 SMD, TMI and CIK.

### **60.4.5 Individual Reports Windows**

In this implementation, seven report message types will be sent from the DMCC to the helicopter simulation entities, including:

<b>SPOT</b>
<b>MTD</b>
<b>SHOT</b>
<b>SPLASH</b>
<b>REQUEST</b>
<b>MOVCMD</b>
<b>FREE TEXT</b>

**TYPES OF REPORTS TRANSMITTED BY THE DMCC**

#### **60.4.6 CIK MENU**

The CIK allows the operator to enter alphanumeric data into the system. on board the actual Comanche aircraft. The functionality of the CIK in the Digital Message/Communications Console will be limited to simulation of the CIK's display on the screen of the X-terminal, but the entry of actual alphanumeric information into the DMCC will take place though the QWERTY keyboard of the X-term. Four other keys on the CIK are implemented, the left and right arrow, the Enter key, and CLEAR ALL.

#### **60.4.7 LOGON**

The LOGON menu will be the initial menu seen by the user upon starting the DMCC software. It will prompt the user for the CEOI (an eight character alphanumeric call sign).

##### **60.4.7.1 LOGON TO NETWORK**

The LOGON TO NETWORK SMD bezel on this screen will allow the user to logon to the Digital Message Communications System. The program will wait for a period of time

##### **60.4.7.2 STAND ALONE OPERATION**

This bezel will allow the user to turn on the DMCC client except that it will not transmit or receive messages.

### **60.4.7.3    SYS MAIN Window**

After the user has selected one of the modes of operation, control will pass back to the SYS MAIN screen.

### **60.4.8       ADDRESS LIST EDIT**

The ADDRESS LIST is the list of CALL SIGNS and GROUPS to which the operator can *send* messages from the various reports screens.

#### **60.4.8.1    Call Signs, Group Names**

It contains 8 possible CALL SIGNS (8 character alphanumeric strings) and GROUP NAMES (also 8 character alphanumeric strings).

#### **60.4.8.2    Initial Members**

Initially, the list will have no members. Up to 8 members can be added to the list.

#### **60.4.8.3    ADDRESS LIST**

The LIST is a round-robin tabulated menu of the current GROUPS and CALL SIGNS to which the console can send messages.

#### **60.4.8.4    Adding and Deleting Addresses**

This menu allows the operator to ADD and DELETE various CALL SIGNS and GROUP NAMES from this list, using the TEXT WIDGET, the ADD key, and the DELETE key.

#### **60.4.8.5    ADDRESS TEXT WIDGET**

The text widget is used for entering addresses for addition to the list.

#### **60.4.8.6 SIZING**

An 8 character text widget allows the entry of new CALL SIGNS or GROUP NAMES . Its contents are cleared when the ADD function is activated.

#### **60.4.8.7 UP BUTTON**

The UP button moves the highlight up the LIST, round-robinning to the bottom when the top is reached. If there is 0 or 1 CALL SIGN or GROUP NAME in the list, it does nothing.

#### **60.4.8.8 DOWN BUTTON**

The DOWN button moves the highlight down the LIST, round-robinning to the top when the bottom is reached. If there is 0 or 1 CALL SIGN or GROUP NAME in the list, it does nothing.

#### **60.4.8.9 ADD BUTTON**

The ADD button adds the new CALL SIGN or GROUP NAME from the TEXT WIDGET to the LIST which is displayed in the round-robin selectable list. The ADD button will only add a text string to the list if the string contains a character other than the space character or the null character.

#### **60.4.8.10 DEL BUTTON**

The DEL key deletes the selected CALL SIGN or GROUP NAME from the ADDRESS LIST.

#### **60.4.9 CEOI/GROUP LIST EDIT**

The GROUP LIST is the list containing the individual CEOI or CALL SIGN of the operator, plus up to 7 GROUP NAMES for groups of which the operator is a member (both are 8 character alphanumeric strings).

#### **60.4.9.1 MESSAGE RECEPTION**

The station will *receive* messages addressed to the **CALL SIGN** of the station and messages sent to groups to which the operator claims membership.

#### **60.4.9.2 EDITING THE LIST**

This screen allows the operator to edit the list of **CALL SIGNS** and **GROUP NAMES** for messages he or she wishes to *receive*.

#### **60.4.9.3 FUNCTIONAL DESCRIPTION**

This screen is functionally identical to the **ADDRESS LIST EDIT** menu, except that it manages a separate list of names.

#### **60.4.10 LOCATION LIST EDIT**

The **LOCATION LIST** is the list containing alphanumeric codes of up to 8 characters for up to 8 battlefield locations for use in the **MOVCMD** report.

#### **60.4.10.1 FUNCTIONAL DESCRIPTION**

This screen is functionally identical to the **ADDRESS LIST EDIT** menu, except that it manages a separate list of names.

#### **60.4.11 SYS MAIN**

##### **60.4.11.1 NEW MESSAGE Indicator**

When the DMCC receives a new message, an alert box will pop up on the screen and will announce the type, sender and priority of the incoming message. The warnings will be different depending on the priority of the message. The alert box will be dismissed by the operator by clicking a box in the alert labelled **DISMISS**. The **MESGS** indicator will show the number of messages (that have not been read) in the **IN-BOX**, the type of message just received (according the

table immediately below this paragraph) and its priority (U for Urgent, R for Routine). The MESGS indicator is the number which appears in the envelope.

<b>Receive Message Types</b>	<b>Message Identifier</b>
<b>SPOT</b>	<b>SPT</b>
<b>STATUS</b>	<b>STA</b>
<b>MOVEMENT</b>	<b>MVT</b>
<b>BDA</b>	<b>BDA</b>
<b>LZ/PZ RECON</b>	<b>LZR</b>
<b>AIR ROUTE RECON</b>	<b>ARR</b>
<b>GROUND ROUTE RECON</b>	<b>GRR</b>
<b>BP/OP RECON</b>	<b>BPR</b>
<b>CROSSING RECON</b>	<b>CRR</b>
<b>BRIDGE RECON</b>	<b>BRR</b>
<b>FREE TEXT</b>	<b>FTX</b>
<b>ARTY REPEAT</b>	<b>ARP</b>
<b>ARTY CANCEL</b>	<b>ACL</b>
<b>ARTY CHECK</b>	<b>ACH</b>
<b>ARTY CNO</b>	<b>ACN</b>
<b>ARTY SHIFT</b>	<b>ASH</b>
<b>ARTY NEW MSN</b>	<b>ANM</b>
<b>ARTY END MSN</b>	<b>ANM</b>
<b>REQUEST REPORT</b>	<b>RQR</b>
<b>NBC</b>	<b>NBC</b>
<b>SPLASH</b>	<b>SPS</b>
<b>MIJI</b>	<b>MIJ</b>
<b>NBC</b>	<b>NBC</b>
<b>PIREP</b>	<b>PRP</b>
<b>DNAV</b>	<b>DNV</b>

#### 60.4.11.2      **SYS MAIN TMI BUTTONS**

The **SYS MAIN** menu will have TMI buttons to transfer control to the **ADDRESS LIST** screen and the **CEOI/GROUP LIST** screen. It will have a soft bezel to transfer control to the **SYS MAIN MSG** screen.

It will also contain a button to **LOGOFF**.



## **60.5 SYS MAIN MSG1 Window**

When the DMCC operator wants to read a message he or she has just received, or dispose of a message he or she has just read, or do something with a message he or she received some time ago and had kept, the operator will start with the **MESSAGES MAIN MENU**.

### **60.5.1 THE IN-BOX**

A numbered list of all the messages in the **IN-BOX** shall be displayed, with the message type, the sender, the time the message was sent, and the priority given for each message. The list shall be arranged first by priority and second by time of message receipt, with the most urgent messages appearing in a group at the top of the list. Urgent messages shall be marked with a **U**, while routine priority messages shall be marked with an **R**. Message priority shall be determined by the sender of the message. Upon entering the **MESSAGES** main menu, the top message in the list shall be highlighted.

### **60.5.2 ENVELOPE ICON**

There shall be an **ENVELOPE ICON** at the bottom of the simulated **SMD** window that contains the **MSG** indicator which indicates the number of new messages that have not been read. This icon shall appear when a new message is received by the **DMCC**. When a message has been read but has not been disposed of, it shall still exist as an active message in the **IN-BOX** but will not be reflected in the number on the envelope.

### **60.5.3 MSG READ**

Pressing **READ** shall change the window from the list of available messages to the display of the contents of the highlighted message in the **Sys Main Mesg Read** screen. When **READ** is **ON**, a second press of the **READ** button will dismiss the contents and return to the list of messages available. This option shall only be available if messages are available to be acted upon. The second line of the **CIK** shall be blank until **READ** is on at which time the second line shall read **"ON"**.

### **60.5.4 Message Display Formats**

Display formats for the information contained in the various reports are textual information, displayed in the **SYS MAIN MSG READ** submenu. They are found at the end of this document.

#### **60.5.5 PRE MSG and NEXT MSG Bezels**

If the operator decides to read a message in the IN BOX, he or she will use the PRE MSG and the NEXT MSG bezels to move the highlight onto the desired message, then will press READ MSG. Upon entering the READ MSG function the contents of the message highlighted on the MESSAGES main menu are displayed on the screen.

#### **60.5.6 DELETE MSG TMI**

Pressing the DELETE MSG TMI shall delete a message from the IN-BOX . This option shall only be available if messages are available to be acted upon.

#### **60.5.7 SAVE & EXIT**

The SAVE & EXIT switch shall save any changes made while in the MSG menu and shall return control to the SYS MAIN menu.

#### **60.5.8 RE-USE**

Pressing the RE-USE TMI shall allow the operator to forward the same message to someone else. Control will transfer to the REPORTS menu, from where a new message may be appended to the original message.

#### **60.5.9 REPLY**

When a message is selected from the MAILBOX, a REPLY TMI will become active. This switch will allow the operator to transfer to the REPORTS menu, where he will be able to generate a new message of preformatted or free-text type, to be sent to the sender of the original message. The ADRS bezel will already have selected the sender of the first message as the addressee of the new message to be sent back.

**60.5.10      PREV, NEXT**

Pressing the **PREV** bezel and the **NEXT** bezel shall move the highlight up and down the list of messages in the **IN-BOX** to select a message on which to perform some action. The highlight shall jump to the top message when the **NEXT** bezel is pressed and the highlight is on the bottom most message. Likewise, when the topmost message is highlighted and the **PREV** bezel is pressed, the highlight shall move to the bottom-most item on the list. This option shall only be available if messages are available to be acted upon.

## **60.6 RPRT Windows**

The **RPRT MENU** is the master menu for sending messages (called **REPORTS**) from the **DMCC** to the other **DMC**-capable simulation entities in the **AIRNET** simulation facility or elsewhere in the network. The following paragraphs describe the various widgets in this menu and their uses and effects.

### **60.6.1 SAVE & EXIT**

This switch shall save any changes made while in the **RPRT MENU** and shall return to the **SYS MAIN MENU** or whatever function was being used when **RPRT** was invoked.

### **60.6.2 SHOT**

Upon selection of this switch, the system shall display the **SHOT TMI's** and **SMD** page. (see below).

### **60.6.3 SPLASH**

Upon selection of this switch, the system shall display the **SPLASH TMIs** and **SMD** page (see below).

### **60.6.4 MTO**

Upon selection of this switch, the system shall display the **MTO TMIs** and **SMD** page (see below).

### **60.6.5 SPOT**

The **SPOT** report is implemented in the **DMCC** even though no ground person is anticipated to want to send a **SPOT** report. This is to allow the assessment of the **DMCC** architecture for in-cab simulated communications, and to provide a simple means of integration testing. Upon selection of this switch, the system shall display the **SPOT TMIs** and **SMD** page (see below).

#### **60.6.6 MOVCMD**

Upon selection of this switch the DMCC shall display the MOVCMD TMI's and the MOVCMD SMD page.

#### **60.6.7 REQ**

Upon selection of this REQUEST switch the DMCC shall display the RPRT REQ TMI's and the REQ SMD page.

#### **60.6.8 FREE TXT**

Upon selection of this switch the DMCC shall display the RPRT FREE TXT TMI's and the FREE TXT SMD page.

## **60.7 RPRT Window Functionality**

The following sections detail the screens which are used to send messages from the DMCC. Each screen shares the following features:

### **60.7.1 COMMON RPRTS WINDOW FUNCTIONS**

#### **60.7.1.1 ADRS**

Once the screen function is entered, the DMCC operator shall select an addressee using the ADRS soft bezel and the MORE bezel. This will take the form of a tape type list of 6 possible addressees. The addressees will be listed in boxes on this tape, and the operator will use the ADRS key to cycle a highlight along this tape., wiht the highlight round robining to the beginning when the key is pressed while the highlight is at the end. The MORE key will be used to display the remaining addressees if there are more than 6, and will also be used to go back to the first set of selections when the remaining ones are being displayed.

#### **60.7.1.2 TAPE WIDGET**

This round-robin tape widget will be used to perform screen specific choices also, as outlined below.

#### **60.7.1.3 TEXT MSG**

Pressing the TEXT MSG TMI shall allow the DMCC operator to add any comments to the report, by entering it on the KEYBOARD and pressing the KEYBOARD RETURN key or clicking on the ENTER key in the CIK. If no text message is entered, the text message display area of the screen shall remain blank.

#### **60.7.1.4 SEND ROUTIN**

The SEND ROUTIN TMI shall operate as described in the SPOT REPORT section.

**60.7.1.5 SEND URGENT**

The **SEND URGENT** TMI shall operate as described in the **SPOT REPORT** section.

**60.7.1.6 CLR & RETURN**

The **CLR & RETURN** TMI shall terminate the report function without saving any changes made and the display shall return to the **SYS MAIN RPRT** menu.

**60.7.1.7 SAVE & RETURN**

Pressing the **SAVE & RETURN** shall save any changes made while in the current function and the display shall return to the **RPRT** menu.

**60.7.1.8 ENUMERATED TYPES DEFAULT VALUE**

If a selection is not made for any of the selection fields in any of the reports, then no highlight will appear on the SMD. (In this case, enumerated type 0 for that type will be transmitted, meaning NOT ENTERED. In the case where a message is received containing NOT ENTERED for a field, the message read display for that field shall be blank.

**60.7.2 SPOT REPORT**

The **SPOT** menu will have round-robin highlight menu choices for the following:

**ENEMY TYPE (ADA, SAM, TANK, WHLD, TRKD, ACFT, TRPS, UNCL)**

**ENEMY ACTIVITY (STATIONARY, MOVING, DUG-IN, RETREATING, ATTACKING DMGD, KILLED, MOBIL\_KILL);**

**DIRECTION (N, NE, E, SE, S, SW, W, NW);**

**OBS INT (CONT\_MSN, ENGA, RTS IHLD, OBSV TGT).**

**60.7.2.1 Enemy Speed**

Text fields for entry of enemy **SPEED** (3 char) and **UNIT** (8 char) will be provided.

**60.7.2.2 Text Msg TMI**

The **TEXT MSG TMI** will allow text to be entered in the 23 character **CIK** test widget for editing free text to be included within the message.

**60.7.2.3 MPH/KPH Switch**

A TMI toggle switch shall select between **MPH** and **KPH**.

**60.7.2.4 Enemy Type**

Each time the **ENEMY TYPE** button is pressed, the GUI software will make a call to the Date Time Group routine and record the **DTG**. This **DTG** represents the **EFFECTIVE TIME** or **TIME SIGHTED**, and will be passed as an argument to the **SPOT PDU BUILDER** when **SEND** occurs.

**60.7.2.5 Enemy Number**

A 3 character decimal integer field shall be provided for entering the **NUMBER** of the enemy.

**60.7.2.6 Enemy Location**

The lower **CIK** text widget shall be labeled **LOCN** and will be used for entry of a UTM coordinate of up to 16 alphanumeric characters.

**60.7.3 MTO**

The **MTO** menu has three **TMIs** to select pre-formatted messages for transmission.



#### **60.7.3.1 REQ ADJ**

The REQ ADJ TMI will select a REQUEST ADJUST message for transmission.

#### **60.7.3.2 EAT**

The EAT? TMI will select an ENTER AS TARGET? message for transmission.

#### **60.7.3.3 EOM**

The EOM? TMI will select an END OF MISSION? message for transmission.

#### **60.7.3.4 TEXT MSG**

The TEXT MSG TMI shall operate as described in the SPOT REPORT section.

### **60.7.4 SHOT**

The SHOT report menu involves no buttons, other than the standard ones which all message screens use as described above. It sends the SHOT message and then displays the words SHOT REPORT SENT for 2 seconds in a small pop up window, and returns to the calling menu.

### **60.7.5 SPLASH**

The SPLASH report menu sends the SPLASH message and then displays the words SPLASH REPORT SENT for 2 seconds in a pop up window, and returns to the calling menu.

### 60.7.6 MOVCMO

The Comanche Movement report shall be used to direct units to a particular location to perform a particular activity, or to inform others of one's intent to move somewhere. It is intended to replace the traditional verbal or written FRAGO (Fragmentary Order) which has been used to convey timely changes to missions which were previously outlined in OPORDs (Operational Orders). The MOVCMO report shall also be used to hand over targets to a team member for servicing when speedy message composition is not a factor.

#### 60.7.6.1 TASK

The operator DMCC shall select in a tape widget the desired movement task using the TASK soft bezel and the MORE bezel exactly as prescribed for selecting an addressee. The options under TASK shall include "MOV TO", "HOLD AT", "CONT MSN", "RENDZ AT", "ENGAGE TGT AT", "MVNG TO", "HLDNG AT", "ARVING AT", "PSNG THRU", and "DPRTING FROM".

#### 60.7.6.2 When

Pressing the WHEN bezel shall round-robin a highlight among the options given for when the movement will take place. Options shall include "IMMED", "WHEN RDY", "AMC", and "TIME". When the highlight is moved to the last option, "TIME", the alphanumeric KEYBOARD shall be enabled for text input of a time. The time is entered in the form

hh:mm L

or

hh:mm Z.

L stands for Local Time and Z stands for Zulu, or Greenwich Mean Time, the time at the Prime Meridian in Greenwich, England.

The simulated CIK shall display the prompt "TIME?" on the first line, with the operator entering the time on the second line.

If the operator entered the time in Local Time, the DMCC must change the time to ZULU time prior to transmission. This is done by accessing a file containing

the ZULU OFFSET, which is the number of minutes Local Time is AHEAD of ZULU time. This file is changed whenever the timezone of the DMCC changes.

#### **60.7.6.3      Location**

Pressing the LOCN bezel shall round-robin a highlight among the options given for where the movement will take place. Options shall include "MY POSN", and those positions entered in the location list edit screen.

#### **60.7.7    REQT REPORT**

The REQUEST REPORT function shall be used when the DMCC operator wants to ask another DMC entity to send him a specific kind of report. Figure 6.6.5.22 outlines the REQT REPT.

##### **60.7.7.1      Report Type Selection**

A round robin horizontal tape selection of the type of report to be requested shall be presented. The options shall include

NONE  
STATUS  
SPOT  
RECON  
MOVEMENT  
PIREP  
MIJI

##### **60.7.7.2      Recon Type Selection**

When the Recon report type is selected, another horizontal tape selection for the type of recon report desired. The options shall include:

GND RTE  
AIR RTE  
BRDG  
LZ/PZ  
BP/OP  
CRSG

#### **60.7.8 FREE TEXT**

The **FREE TEXT** message window will basically be a 256 character editable text widget.

**Note: These drawings provide row and column numbers which are not displayed on the actual message read window. Dots are also not displayed; they are provided here for clarity of character position.**

```

0      1      2      3
01234567890123456789012345678901234

0  .SPOT MESSAGE.....*URGENT*.....
1  .....
2  .SENDER .....XXXXXXXXX.....
3  .SENT TO.....XXXXXXXXX.....
4  .FWD BY.....XXXXXXXXX.....
5  .MSG SENT.....26 1745 JUN 95..
6  .XMIT LCN.. .....NB 0624 0332...
7  .XMIT ALT.....12000 FEET.....
8  .....
9  .GT QUAN.....nnnn.....
10 .GT TYPE.....TRKD.....
11 .TGT ACTIV.....MVNG.....
12 .TGT SPEED.....20 MPH.....
13 .TGT DIRN.....NW.....
14 .TGT UNIT.....aaaaaaaaaaaaaaaaa
15 .TGT LCN.....NB 2322 2323...
16 .TIME TGT SGHTD...26 1744 JUN 95..
17 .OBS LCN AT CNTCT..NB 1233 4333..
18 .OBS INTENT.....ENGA.....
19 .....
20 .FREE TEXT ANNOTATION*.....

```

## FREE TEXT REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.FREE TEXT MESSAGE.ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
10	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
11	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
12	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
13	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
14	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
15	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
16	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
17	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
18	...aaaaaaaaaaaaaaaaaaaaaaaaaaaa.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

## ARTILLERY CANCEL REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY CANCEL.RPRT..ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.....		
10	.....		
11	.....		
12	.....		
13	.....CANCEL.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

March 31, 1993

## ARTILLERY REPEAT REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY REPEAT REPT..ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.....		
10	.....		
11	.....		
12	.....		
13	.....REPEAT.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION•••.....		



0 1 2 3  
01234567890123456789012345678901234

Page 592

## ARTILLERY CNO REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY CNO REPORT...ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXX.....		
3	.SENT TO.....XXXXXXXX.....		
4	.FWD BY.....XXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.MISSION ID.....AAAAAAAAAAAAAAAA		
10	.TARGET ID.....AAAAAAAAAAAAAAAA		
11	.MISSION STATUS.....AAAAAAAAAAAAA...		
12	.FIRE FOR EFFECT...NO.....		
13	.....		
14	.....		
15	.....CAN NOT OBSERVE.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

```

0      1      2      3
01234567890123456789012345678901234
0 .ARTY SHIFT REPORT.ROUTINE.....
1 .....
2 .SENDER .....XXXXXXXXX.....
3 .SENT TO.....XXXXXXXXX.....
4 .FWD BY.....XXXXXXXXX.....
5 .MSG SENT.....26 1745 JUN 95..
6 .XMIT LCN.....NB 0624 0332....
7 .XMIT ALT.....1200 FEET.....
8 .....
9 .MISSION ID.....AAAAAAAAAAAAAAAA
10 .TARGET ID.....AAAAAAAAAAAAAAAA
11 .MISSION STATUS...AAAAAAAAAAAAA...
12 .FIRE FOR EFFECT..NO.....
13 .....
14 .....
15 .....SHIFT.....
16 .....
17 .....AAAAAAAAAAAAAAAA.....
18 .....
19 .....
20 .FREE TEXT ANNOTATION***.....

```

March 31, 1993

## ARTILLERY NEW MISSION REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY NEW MSN RPRT.ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	... * * NEW ARTILLERY MISSION * *...		
10	.....		
11	.MISSION ID.....AAAAAAAAAAAAAAAA		
2	.TARGET ID.....AAAAAAAAAAAAAAAA		
13	.MISSION STATUS.....AAAAAAAAAAAAA...		
14	.MISSION TYPE.....IMMED SUPPR.....		
15	.SHELL.....WP.....		
16	.CONTROL.....WR.....		
17	.FUZE.....TIME DELAY.....		
18	.TRAJ.....LOW.....		
19	.FIRE FOR EFFECT...YES.....		
20	.FREE TEXT ANNOTATION***.....		

## ARTILLERY MESSAGE TO OBSERVER REPORT

0	1	2	3
01234567890123456789012345678901234			

0 .ARTY MTO REPORT...ROUTINE.....  
1 .....  
2 .SENDER .....XXXXXXXX.....  
3 .SENT TO.....XXXXXXXX.....  
4 .FWD BY.....XXXXXXXX.....  
5 .MSG SENT.....26 1745 JUN 95..  
6 .XMIT LCN.....NB 0624 0332....  
7 .XMIT ALT.....1200 FEET.....  
8 .....  
9 .MISSION ID.....AAAAAAAAAAAAAAAAA  
10 .TARGET ID.....AAAAAAAAAAAAAAAAA  
11 .MISSION STATUS.....AAAAAAAAAAAAA..  
12 .....  
13 .....MESSAGE TO OBSERVER.....  
14 .....  
15 .REQUEST ADJUST...YES.....  
16 .ENTER AS TARGET...NO.....  
17 .END MISSION.....NO.....  
18 .....  
19 .....  
20 .FREE TEXT ANNOTATION\*.....

March 31, 1993

## ARTILLERY SHOT REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY SHOT REPORT..ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.MISSION ID.....AAAAAAAAAAAAAAAA		
10	.TARGET ID.....AAAAAAAAAAAAAAAA		
11	.MISSION STATUS.....AAAAAAAAAAAAA...		
12	.....		
13	.....		
14	.....		
15	.....* * SHOT FIRED * *.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION... ..		

```

0      1      2      3
01234567890123456789012345678901234
0 .ARTY SPLASH RPRT..ROUTINE.....
1 .....
2 .SENDER .....XXXXXXXXX.....
3 .SENT TO.....XXXXXXXXX.....
4 .FWD BY.....XXXXXXXXX.....
5 .MSG SENT.....26 1745 JUN 95..
6 .XMIT LCN.....NB 0624 0332....
7 .XMIT ALT.....1200 FEET.....
8 .....
9 .MISSION ID.....AAAAAAAAAAAAAAAA
10 .TARGET ID.....AAAAAAAAAAAAAAAA
11 .MISSION STATUS...AAAAAAAAAAAAA...
12 .....
13 .....
14 .....* * SPLASH * *.....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .FREE TEXT ANNOTATION...

```

## ARTILLERY END OF MISSION REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.ARTY EOM REPORT...ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.MISSION ID.....AAAAAAAAAAAAAAAA		
10	.TARGET ID.....AAAAAAAAAAAAAAAA		
11	.MISSION STATUS.....AAAAAAAAAAAAA...		
12	.....		
13	.....* * END OF MISSION * *.....		
14	.....		
15	.DISPOSITION.....BURNING.....		
16	.RECORD AS TARGET..NO.....		
17	.CASUALTIES.....3000.....		
18	.POINT NUMBER.....AAAAAAAAAAAAAAAA.		
19	.....		
20	.FREE TEXT ANNOTATION*.....		



March 31, 1993

## NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 1 REPORT

0	1	2	3
0	1	2	3
01234567890123456789012345678901234			
0	.NBC-1 REPORT.....URGENT.....		
1	.....		
2	.SENDER .....XXXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.....>>>> NUCLEAR BLAST <<<<.....		
10	.....		
11	.DESCRIPTION.....INCREASING.....		
12	.BURST TYPE.....SURFACE.....		
13	.DELIVERED BY.....ROCKET.....		
14	.CLOUD HGHT UNITS..DEGREES.....		
15	.CLOUD HEIGHT.....35.....		
16	.CLOUD DESCRIPTION.AAAAAAAAAAAAAAAAAA		
17	.FLASH TO BANG.....55.SECONDS.....		
18	.START TIME.....26 1743 JUN 95..		
19	.STOP TIME.....26 1744 JUN 95..		
20	.FREE TEXT ANNOTATION*.....		

## NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 4 REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.NBC-4 REPORT.....URGENT.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.*.CHEMICAL OR BIOLOGICAL ATTACK.*.		
10	.....		
11	.DESCRIPTION.....INCREASING.....		
12	.BURST TYPE.....SURFACE.....		
13	.DELIVERED BY.....ROCKET.....		
14	.CLOUD HGHT UNITS..DEGREES.....		
15	.CLOUD HEIGHT.....35.....		
16	.CLOUD DESCRIPTION.AAAAAAAAAAAAAAAAAA		
17	.DOSE RATE.....300.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

March 31, 1993

## NUCLEAR, BIOLOGICAL, CHEMICAL TYPE 5 REPORT

0	1	2	3
01234567890123456789012345678901234			

0	.NBC-5 REPORT.....ROUTINE.....
1	.....
2	.SENDER .....XXXXXXXXX.....
3	.SENT TO.....XXXXXXXXX.....
4	.FWD BY.....XXXXXXXXX.....
5	.MSG SENT.....26 1745 JUN 95..
6	.XMIT LCN.....NB 0624 0332....
7	.XMIT ALT.....1200 FEET.....
8	.....
9	.....
10	.....
11	.....
12	.....
13	.....NBC NEGATIVE.....
14	.....
15	.....
16	.....
17	.....
18	.....
19	.....
20	.FREE TEXT ANNOTATION***.....

## STATUS REPORT

0	1	2	3
01234567890123456789012345678901234			
0	. STATUS REPORT.....	ROUTINE.....	
1	.....		
2	. SENDER .....	XXXXXXXX	
3	. SENT TO.....	XXXXXXXX	
4	. FWD BY.....	XXXXXXXX	
5	. MSG SENT.....	26 1745 JUN 95..	
6	. XMIT LCN.....	NB 0624 0332....	
7	. XMIT ALT.....	1200 FEET.....	
8	.....		
9	. FUEL.....	XXXXX LBS.....	
10	. FAILED EQUPT.....	aaaaaaaaaaaaaaaa	
11	. FAILED EQUPT.....	aaaaaaaaaaaaaaaa	
12	. FAILED EQUPT.....	aaaaaaaaaaaaaaaa	
13	. HELLFIRES.....	XX.....	
14	. STINGERS.....	XX.....	
15	. ROCKETS.....	XX.....	
16	. ROUNDS.....	XXXX.....	
17	. REQUEST TYPE.....	AUTOMATIC.....	
18	.....		
19	.....		
20	. FREE TEXT ANNOTATION...		

**March 31, 1993**

**REQUEST REPORT: STATUS, SPOT, MOVEMENT**

```

0      1      2      3
01234567890123456789012345678901234
0 .REQUEST REPORT...ROUTINE.....
1 .....
2 .SENDER .....XXXXXXXXX.....
3 .SENT TO.....XXXXXXXXX.....
4 .FWD BY.....XXXXXXXXX.....
5 .MSG SENT.....26 1745 JUN 95..
6 .XMIT LCN.....NB 0624 0332....
7 .XMIT ALT.....1200 FEET.....
8 .....
9 .REPORT DESIRED...STATUS.....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .FREE TEXT ANNOTATION*.....

```

```

0      1      2      3
01234567890123456789012345678901234

0 .REQUEST REPORT....ROUTINE.....
1 .....
2 .SENDER .....XXXXXXXXX.....
3 .SENT TO.....XXXXXXXXX.....
4 .FWD BY.....XXXXXXXXX.....
5 .MSG SENT.....26 1745 JUN 95..
6 .XMIT LCN.....NB 0624 0332...
7 .XMIT ALT.....1200 FEET.....
8 .....
9 .REPORT DESIRED....RECON.....
10 .TYPE DESIRED.....AIR ROUTE.....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .FREE TEXT ANNOTATION*.....

```

March 31, 1993

## BATTLE DAMAGE ASSESSMENT REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.BDA REPORT.....	ROUTINE.....	
1	.....		
2	.SENDER .....	XXXXXXXX	
3	.SENT TO.....	XXXXXXXX	
4	.FWD BY.....	XXXXXXXX	
5	.MSG SENT.....	26 1745 JUN 95..	
6	.XMIT LCN.....	NB 0624 0332....	
7	.XMIT ALT.....	1200 FEET.....	
8	.....		
9	.STRIKE START.....	26 1740 JUN 95..	
10	.STRIKE END.....	26 1742 JUN 95..	
11	.TARGET CATEGORY.....		
12	.TARGET TYPE.....	TANK.....	
13	.TARGETS DESTROYD...	4.....	
14	.PERCENT COVERAGE..	100.....	
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION...		

March 31, 1993

## GROUND ROUTE RECON REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.GND RTE RECON RPT.ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.ENEMY ACTIVITY....ATTACKING.....		
10	.CLSFACTN FORMULA...MLC.....		
11	.CLSFACTN INFO.....AAAAAAAAAAAAAAAA		
12	.ROUTE ID.....AAAAAAAAAAAAAAAA		
13	.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		



March 31, 1993

## AIR ROUTE RECON REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.AIR RTE RECON RPT.ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.ENEMY ACTIVITY....ATTACKING.....		
10	.....		
11	.OBSTACLES.....TREES.....		
12	.....		
13	.ROUTE ID.....AAAAAAAAAAAAAAAA		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION... ..		

## BRIDGE RECON REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.BRIDGE RECON RPT..ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.BRIDGE TYPE.....GIRDER.....		
9	.DAMAGE.....SEVERE.....		
10	.SPANS.....6.....		
11	.CONST MATL.....METAL.....		
12	.LENGTH.....AAAA.....		
13	.WIDTH.....AAAA.....		
14	.HEIGHT.....AAAA.....		
15	.UNDER.....AAAA.....		
16	.CONSTR DESCR.....AAAAAAAAAAAAAAAA		
17	.ID.....AAAAAAAAAAAAAAAA		
18	.SPAN LENGTH.....AAAA.....		
19	.LOAD CLASS.....AAAA.....		
20	.FREE TEXT ANNOTATION***.....		

Note: Only first 16 chars in 32 char  
PDU field used for bridge ID

March 31, 1993

## LANDING ZONE/PICKUP ZONE RECON REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.LZ/PZ RECON RPT. .ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXX.....		
3	.SENT TO.....XXXXXXXX.....		
4	.FWD BY.....XXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.ACTIVITY LIKELY...EXPECTED.....		
10	.OBSTACLES.....TWR/ANT.....		
11	.OBST. DESCR.....AAAAAAAAAAAAAAAA		
12	.LZ/PZ ID.....AAAAAAAAAAAAAAAA		
13	.LZ/PZ SIZE.....AAAAAAAAAAAAAAAA		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

March 31, 1993

## BP/OP RECON REPORT

	0	1	2	3
	01234567890123456789012345678901234			
0	.BP/OP RECON RPT...ROUTINE.....			
1	.....			
2	.SENDER .....XXXXXXXXX.....			
3	.SENT TO.....XXXXXXXXX.....			
4	.FWD BY.....XXXXXXXXX.....			
5	.MSG SENT.....26 1745 JUN 95..			
6	.XMIT LCN.....NB 0624 0332....			
7	.XMIT ALT.....1200 FEET.....			
8	.....			
9	.ENEMY ACTIVITY...EXPECTED.....			
10	.OBSTACLES.....WIRES.....			
11	.OBST. DESCR.....AAAAAAAAAAAAAAAA			
12	.BP/OP ID.....AAAAAAAAAAAAAAAA			
13	.BP/OP SIZE.....AAAAAAAAAAAAAAAA			
14	.AXIS.....AAAAAAAAAAAAAAAA			
15	.....			
16	.....			
17	.....			
18	.....			
19	.....			
20	.FREE TEXT ANNOTATION***.....			

March 31, 1993

## CROSSING REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.CROSSNG RECON RPT.ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.BANK SLOPE ENTRY..AAAA.....		
10	.BANK SLOPE EXIT...AAAA.....		
11	.CROSSING LENGTH...AAAA.....		
12	.CROSSING WIDTH...AAAA.....		
13	.CROSSING DEPTH...AAAA.....		
14	.CURRENT FLOW.....AAAA.....		
15	.CROSSING ID.....AAAAAAA.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

March 31, 1993

## DOWNED AIR VEHICLE REPORT

0	1	2	3
0	1	2	3
01234567890123456789012345678901234			
0	.DNAV REPORT.....URGENT.....		
1	.....		
2	.SENDER .....XXXXXXXX.....		
3	.SENT TO.....XXXXXXXX.....		
4	.FWD BY.....XXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.AIRCRAFT TYPE.....OH-58.....		
10	.AIRCRAFT STATUS...RECOVER.....		
11	.PILOT STATUS.....GOOD.....		
12	.....		
13	.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

## Movement Command (MOVCMD) REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.MOVCMD REPORT,....URGENT.....		
1	.....		
2	.SENDER .....XXXXXXXX.....		
3	.SENT TO.....XXXXXXXX.....		
4	.FWD BY.....XXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.TARGET ID.....AAAAAAAAAAAAAAAA		
10	.TASK.....CONT MSN.....		
11	.WHO.....B3.....		
12	.WHEN.....1245 Z; 0445 L..		
13	.WHERE.....AAAAAAA.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

## PIREP REPORT

0	1	2	3
01234567890123456789012345678901234			

0 .PILOT REPORT.....ROUTINE.....  
1 .....  
2 .SENDER .....XXXXXXXXX.....  
3 .SENT TO.....XXXXXXXXX.....  
4 .FWD BY.....XXXXXXXXX.....  
5 .MSG SENT.....26 1745 JUN 95..  
6 .XMIT LCN.....NB 0624 0332....  
7 .XMIT ALT.....1200 FEET.....  
8 .....  
9 .VISIB..HALF MILE..BLIZZARD.....  
10 .CLDS SCTRD, BASE 12000, TOP.30000.  
11 .....  
12 .TEMP.. -5.C.....BAR 29.56".....  
13 .....  
14 .WIND 80 KNOTS, NORTH EAST.....  
15 .....  
16 .CONT, EXTREME TURBULENCE.....  
17 .....  
18 .SEVERE CLEAR ICING.....  
19 .....  
20 .FREE TEXT ANNOTATION... ..



March 31, 1993

## MIJI REPORT

0	1	2	3
01234567890123456789012345678901234			
0	.MIJI REPORT.....ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXX.....		
3	.SENT TO.....XXXXXXXX.....		
4	.FWD BY.....XXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.AFFECTED FREQ.....xxx.xxxxxx kHz..		
10	.PROGRAMD FREQ.1...xxx.xxxxxx kHz..		
11	.PROGRAMD FREQ.2...xxx.xxxxxx kHz..		
12	.PROGRAMD FREQ.3...xxx.xxxxxx kHz..		
13	.PROGRAMD FREQ.4...xxx.xxxxxx kHz..		
14	.START TIME.....26 1711 JUN 95..		
15	.STOP TIME.....26 1711 JUN 95..		
16	.DESCRIPTION.....WARBLING.....		
17	.PERCENT LOST.....XXX PERCENT.....		
18	.TYPE.....JAMMING.....		
19	.....		
20	.FREE TEXT ANNOTATION***.....		

NOTE: Frequencies come in as 64 bit floating point numbers which express cycles per second (Hertz). This is to be changed to kiloHertz, (kHz) if the number is greater than or equal to 1000 and less than a million; it is to be changed to Megahertz (MHz) if the value is greater than or equal to a million and less than a billion; and it is to be changed to GigaHertz (GHz) if it is over a billion. The notations Hz, kHz, MHz, and GHz (case exactly as shown here) are to be used. One to three decimal places are to be to the left of the decimal point; exactly 6 (with right filled zeroes) are to be to the right.

March 31, 1993

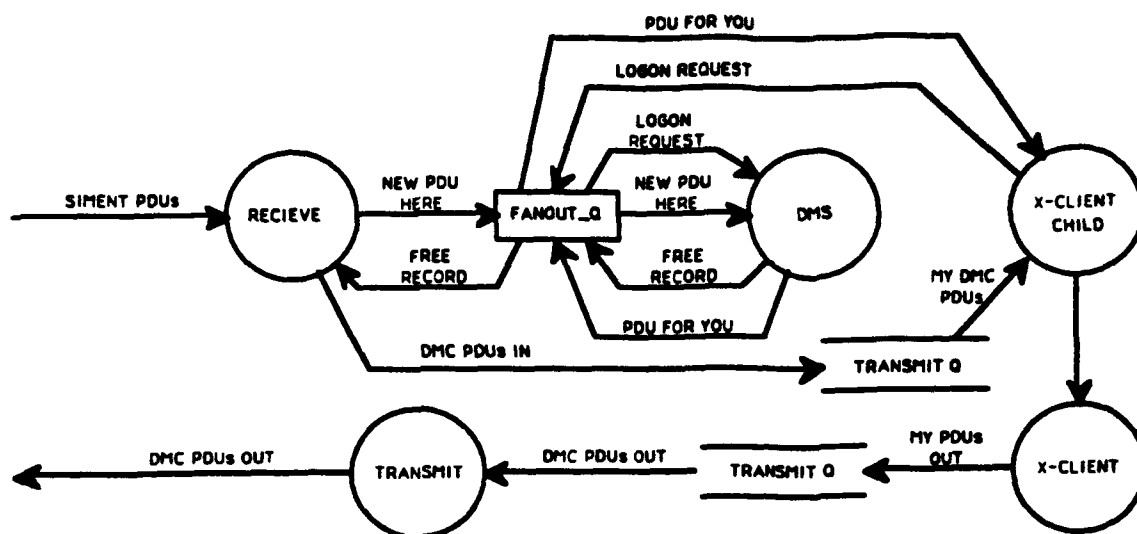
## ACKNOWLEDGEMENT

0	1	2	3
01234567890123456789012345678901234			
0	.ACKNOWLEDGEMENT...ROUTINE.....		
1	.....		
2	.SENDER .....XXXXXXXXX.....		
3	.SENT TO.....XXXXXXXXX.....		
4	.FWD BY.....XXXXXXXXX.....		
5	.MSG SENT.....26 1745 JUN 95..		
6	.XMIT LCN.....NB 0624 0332....		
7	.XMIT ALT.....1200 FEET.....		
8	.....		
9	.YOUR MESSAGE TO...AAAAAAA.....		
10	.WAS RECEIVED AT...26 1743 JUN 95..		
11	.....		
12	.....		
13	.....		
14	.....		
15	.....		
16	.....		
17	.....		
18	.....		
19	.....		
20	.FREE TEXT ANNOTATION••.....		

## 70. Appendix G - Data Flow Diagrams.

The following diagrams contain data flow diagrams for the DMCC System and the DMCC X-Client Process.

### 70.1 Top Level DMCC Inter-Process Data Flow Diagram



### 70.2 DMCC X-Client Process Data Flow Diagram

